UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Sistema para la gestión de datos de redes de sensores multimodales

Yue Dong
Tutor: Irene Rodríguez Luján
Ponente: Pablo Varona Martínez

JUNIO 2017

Sistema para la gestión de datos de redes de sensores multimodales

AUTOR: Yue Dong
TUTOR: Irene Rodríguez Luján

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2017

Resumen

Actualmente existen una gran cantidad de aplicaciones web que ayudan a las personas en sus tareas diarias, satisfaciendo así uno de los objetivos del desarrollo de la tecnología como es reducir el procesamiento manual de tareas automatizables. Además, estos beneficios se ven incrementados en el caso de tareas pesadas y repetitivas que resultan tediosas para el humano mientras que las máquinas pueden solventarlas satisfactoriamente.

Dentro de este marco, el objetivo principal de este Trabajo Fin de Grado es crear una aplicación web que facilite la gestión de los datos recogidos por redes de sensores multimodales instaladas por el Grupo de Neurocomputación Biológica en diferentes ubicaciones de la Escuela Politécnica Superior. Las lecturas de los sensores son recogidas por una versión adaptada del sistema Environmental Sensor Data Repository (ESDR) diseñado por un grupo de investigación de la Universidad Carnegie Mellon. Hasta ahora, la etiquetación de eventos como clases, exámenes, seminarios, etc. para un posterior análisis estadístico de los datos y/o aplicación de técnicas de reconocimiento de patrones se realizaba a través de formularios de Google que requerían de un posterior procesamiento manual relativamente complejo para la incorporación de esta información en el sistema ESDR. Con el objetivo de reducir la carga de trabajo manual en la inyección de datos, mejorar las funcionalidades y experiencia del usuario para la etiquetación de eventos, y facilitar el mantenimiento del sistema global, este Trabajo de Fin de Grado ha creado una aplicación web que (i) facilita la generación de eventos en un calendario y la posterior etiquetación de los mismos; (ii) procesa la información sobre eventos de forma automática y transparente a los investigadores del grupo para su integración directa con ESDR; y (iii) genera matrices de datos de acuerdo a diferentes parámetros especificados por el usuario y con el formato adecuado para un posterior análisis estadístico o uso de algoritmos de aprendizaje automático. La aplicación web desarrollada es sencilla e intuitiva de manejar, es accesible desde cualquier navegador y posee un sistema de login. La herramienta ha sido acogida muy positivamente por los miembros del grupo de investigación al reducir considerablemente la carga de trabajo manual que se venía haciendo hasta ahora.

El desarrollo de esta aplicación se ha realizado en lenguaje Java. Se ha utiliza Wildfly 10 como servidor y MySql como base de datos. Para la implementación de vistas se ha empleado JSF con la librería de *PrimeFaces*. No se ha usado SQL para la manipulación de datos, sino JPA junto con Hibernate.

Este documento contiene todo el proceso de análisis, diseño y desarrollo del proyecto, incluyendo requisitos funcionales, diagrama E-R y diagramas de clase. También presenta las conclusiones sobre el trabajo realizado y posibles mejoras. Como anexo se adjunta un pequeño tutorial para la instalación de la aplicación (Anexo Manual de instalación).

Palabras clave

Narices artificiales, e-nose, Environmental Sensor Data Repository, ESDR, aplicación web, Wildfly, MySql, JSF, PrimeFaces

Abstract

Nowadays there are a lot of web applications that help people in their daily tasks, satisfying one of the goals of technology development such as reducing manual processing of automated tasks. Furthermore, these benefits are increased when dealing with boring and repetitive tasks which are tedious for human, while the machines can solve them satisfactorily.

Within this framework, the main objective of this Bachelor Thesis is the creation of a web application which facilitates the management of data collected by networks of multimodal sensors installed by the Biological Neurocomputation Group in different locations of the Escuela Politécnica Superior. Sensor readings are collected by an adapted version of the Environmental Sensor Data Repository (ESDR) system designed by a research group at Carnegie Mellon University. Until now, the labeling of events such as classes, exams, seminars, etc. for later statistical analysis of the data and/or application of pattern recognition techniques was done through Google Forms which required a complex manual processing to incorporate this information into the ESDR system. With the aim of reducing the manual workload in data injection, improving functionality and user experience for events labeling, and facilitating overall system maintenance, this Bachelor Thesis has created a web application that (i) facilitates the generation of events in a calendar and the subsequent labeling of events; (ii) processes event information automatically and transparently to the researchers for direct integration with ESDR; and (iii) generates data matrices according to different parameters specified by the user and with the appropriate format for statistical and machine learning analysis.

The web application developed is simple and intuitive to use, is accessible from any browser and has a login system. The tool has been welcomed very positively by the members of the research group by considerably reducing the manual workload that was needed so far.

The development of this application has been made in Java. Wildfly 10 has been used as server and MySql as database. JSF with the PrimeFaces library has been used for the implementation of views. SQL has not been used for data manipulation, it has been replaced by JPA & Hibernate.

This document contains the whole details of analysis, design and development of the project, including functional requirements, ER diagram and class diagrams. It also presents the conclusions about the work carried out and further lines of improvement. Finally, a manual for the installation of the application is attached in the annex (Anexo Manual de instalación).

Keywords

Artificial nose, e-nose, Environmental Sensor Data Repository, ESDR, web application, Wildfly, MySql, JSF, PrimeFaces

Agradecimientos

Primero quiero agradecer a mi familia por el apoyo a lo largo de estos años. Sobre todo a mis padres que me han dejado el camino que quería.

También quiero agradecer a mis amigos, que en los últimos años me han dado mucho apoyo y ánimo para que acabara este trabajo.

Gracias a mi tutora Irene, que me ha ayudado y apoyado para poder acabar este proyecto.

Y, por último, gracias a ti, por interesarte en esto.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

Hoy en día existen muchas aplicaciones web que se encargan de gestionar una gran cantidad de datos con determinados objetivos. Con estas aplicaciones, el trabajo para recoger información, agruparla y obtener conclusiones se simplifica en unos sencillos pasos y haciendo clic en ciertos botones. Como ejemplo, podemos pensar en la sección de datos abiertos del Banco Mundial (http://data.worldbank.org/) donde se pueden descargar, visualizar, filtrar y analizar distintos indicadores socioeconómicos y demográficos a diferentes escalas temporales y geográficas.

En este contexto, el Grupo de Neurocomputación Biológica de la EPS tenía la necesidad de manejar de forma sencilla e intuitiva los datos recogidos por sensores instalados en diferentes puntos de la Escuela Politécnica Superior así como la información procedente de anotaciones realizadas por personas externas al grupo de investigación sobre los eventos que tienen lugar en estas ubicaciones. La consecución de este objetivo permitiría generar resultados y conclusiones a partir de esta gran cantidad de datos disponible de forma más eficiente.

El grupo de investigación ya disponía de un sistema de recogida y visualización de datos de sensores ambientales basado en el sistema Environmental Sensor Data Repository (ESDR) [20] desarrollado por la Universidad de Carnegie Mellon. Sin embargo, la interacción con ESDR para integrar los datos procedentes de las anotaciones de eventos resultaba bastante costosa en el día a día ya que se debía generar un fichero JSON con la estructura definida por ESDR. Las anotaciones de eventos se recogían a través de formularios de Google sencillos de configurar por los miembros del grupo y de rellenar por los colaboradores externos. Sin embargo, el uso de estos formularios implicaba transformar datos en formato *Comma-Separated-Value* (CSV) a formato ESDR. El proceso de obtención de matrices de datos para el posterior análisis era similar, pero en sentido contrario: a partir de las descargas de datos desde ESDR en formato CSV o JSON, se generaban unas matrices de datos con *timestamps* equiespaciados (las lecturas de los sensores no tienen por qué serlo) y se ejecutaba un algoritmo para asignar correctamente diferentes etiquetas de anotaciones (por ejemplo, número de personas en un aula) a cada uno de los patrones. Esta tarea se llevaba a cabo a través de la ejecución de varios scripts en Python ejecutados por uno de los miembros del grupo a demanda. La complejidad del proceso de integración y extracción de información hizo que surgiera la necesidad de diseñar e implementar un sistema que hiciera de intermediario entre el sistema ESDR y las distintas funcionalidades requeridas por los miembros del grupo de investigación con el fin de reducir la carga de trabajo manual y aportar transparencia.

Teniendo en cuenta las necesidades del grupo, se optó por desarrollar una aplicación web, debido a las grandes ventajas que presenta respecto a otro tipo de aplicaciones:

- Facilidad de uso: sólo se necesitará conocimientos básicos de informática para trabajar con ella.
- Multiplataforma y acceso concurrente: es accesible desde cualquier tipo de dispositivo y permite el acceso de más de una persona a la vez.
- Ahorro en costes de hardware y software: son accesibles desde un navegador, por lo que no se requiere un dispositivo de alto rendimiento.

1.1 Objetivos

El objetivo de este Trabajo de Fin de Grado es el diseño y desarrollo de una aplicación web que facilita la interacción día a día con el sistema ESDR en base a las necesidades expuestas por los miembros del grupo de investigación involucrados en la recogida y análisis de datos procedentes de las redes de sensores. En particular, los principales objetivos que se quiere conseguir con este trabajo son:

- **Gestión de datos**:
  - Recogida de datos de anotaciones realizadas por profesores externos al grupo de investigación e integración y envío de los mismos al sistema ESDR con el formato correspondiente.
  - Borrado de datos del sistema ESDR.
  - Alta y almacenamiento de horarios y profesores.
- **Generación de estadísticas**: a partir de los datos de ESDR generar informes útiles para el grupo de investigación.
- **Gestión de usuario**: posibilidad de definir distintos tipos de usuario con diferentes permisos sobre la aplicación.

1.2 Organización de la memoria

La memoria consta de los siguientes capítulos:

1. En el capítulo 2 se presenta una visión general de las tecnologías más actuales en desarrollo web y una breve introducción sobre el sistema Environmental Sensor Data Repository (ESDR).

2. En el capítulo 3 se detallan los requisitos del sistema y las herramientas que se va a utilizar el producto final.

3. En el capítulo 4 se define el diseño de la aplicación, como por ejemplo se describen las decisiones que se han tomado para la organización de las vistas, los tipos de datos se guardan en la base de datos, etc.

4. En el capítulo 5 se describen las herramientas y tecnologías utilizadas así como la justificación de las decisiones tomadas en términos de implementación.

5. En el capítulo 6 se especifican las pruebas realizadas y una pequeña presentación del producto final.

6. En el capítulo 7 se realiza una valoración al producto final, incluyendo las posibles mejoras que se puede realizar en el futuro.

2 Estado del arte

Dado que el objetivo de este TFG es desarrollar una aplicación web que se encargará de gestionar datos e interaccionar con otros sistemas, este capítulo tiene como objetivo proporcionar una visión general sobre las nuevas tecnologías en el campo de desarrollo de aplicaciones web. También se incluye una breve presentación del sistema ESDR, encargado de almacenar y visualizar los datos de los sensores y anotaciones.

2.1 Análisis de desarrollo de aplicaciones web

Las aplicaciones web son aquellas herramientas o servicios proporcionados por un servidor web a través de Internet o de una intranet y a los que los usuarios acceden mediante un navegador.

Tener aplicaciones web compactas en sistemas relativamente complejos causa gran cantidad de problemas en términos de escalabilidad, disponibilidad, seguridad, integración, etc. Por ello, con el fin de facilitar la integración de las aplicaciones web en sistemas complejos, éstas se suelen estructurar en tres capas de distinta funcionalidad de acuerdo al diseño Modelo Vista Controlador (MVC) [1] (ver Figura 2-1). Por una parte, la primera capa (capa cliente) la forma el navegador web que se encarga de mostrar la información y definir la forma de presentarla. Por otra parte, la segunda capa corresponde al servicio web, responsable de gestionar las peticiones del usuario (para lo que posiblemente requiera de los servicios de la capa de datos) y mandar los resultados a la capa del cliente. Por último, la última capa se compone típicamente de una base de datos que guarda todos los datos que se necesitan para el correcto funcionamiento de la aplicación.

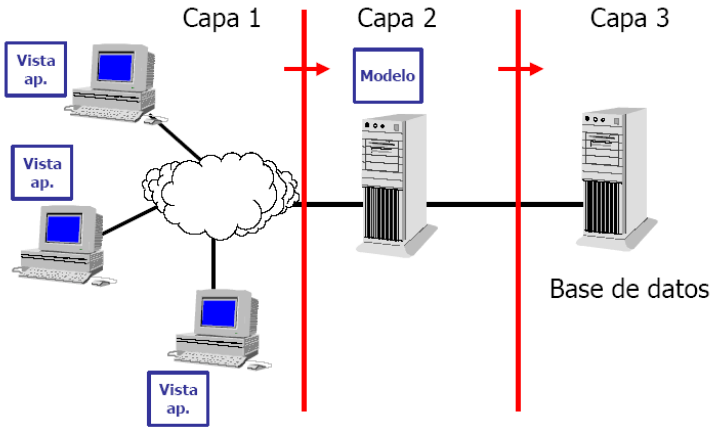


Figura 2-1: Arquitectura en capas [1]

El patrón Modelo Vista Controlador (MVC) separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Esto favorece la reutilización de código y la separación de conceptos, facilita la tarea de desarrollo de aplicaciones así como su posterior mantenimiento.

2.1.1 Tecnologías en el lado cliente

Del lado del cliente se utilizan lenguajes o arquitecturas como HTML o XML que no son propiamente lenguajes de programación. También se utilizan lenguajes interpretados (scripts) para añadir más funcionalidades, especialmente para ofrecer una experiencia interactiva que no requiera recargar la página cada vez. Entre las tecnologías más utilizadas se encuentran: HTML, CSS y JavaScript.

- **HyperText Markup Language (HTML)** [2]: es un estándar que sirve para crear la estructura y el contenido de las páginas web. El código de una página web contiene solamente texto con etiquetas delimitadas por los símbolos < y > que hacen referencia a elementos visuales y externos. Es el navegador o intérprete de código HTML el que se encarga de unir todos estos elementos referenciados y mostrar una página final con imágenes, tablas, videos, etc.

- **Cascading Stylesheets (CSS)** [3]: es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web e interfaces de usuario escritas en HTML o XHTML. También permite aplicar estilos no visuales, como las hojas de estilo auditivas. CSS permite separar claramente el código asociado al contenido en sí de la forma en que éste será presentado, para lo que incluirá características tales como las capas o *layouts* así como los colores y las fuentes a utilizar.

- **JavaScript (JS)** [4]: es un lenguaje de programación interpretado. Se utiliza principalmente en el lado del cliente, es decir se ejecuta directamente sobre un navegador web. Permite mejoras en la interfaz de usuario como efectos de pop up y creación de páginas web dinámicas que muestran diferentes contenidos según la petición del usuario. También existe una forma de JavaScript del lado del servidor, aunque su uso no está tan extendido. Actualmente todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web, por lo que no se necesita ningún intérprete previo a la ejecución.

2.1.2 Tecnologías en el lado servidor

Las tecnologías líderes en la capa del servidor son principalmente JSP, PHP y ASP. El objetivo de todas estas tecnologías es proporcionar contenido dinámico en la web.

- **JavaServer Pages (JSP)** [6]: es una tecnología que permite crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. Se requiere un servidor web compatible con contenedores servlet como Apache Tomcat o Jetty para desplegar y correr JSP. La principal ventaja de JSP frente a otros lenguajes es el uso del lenguaje Java, un lenguaje de propósito general y apto para crear clases que manejen la lógica de negocio y el acceso a datos. Además, es posible ejecutar las aplicaciones en múltiples plataformas sin cambios.
  Hoy en día existe una gran cantidad de *frameworks* de JSP que facilitan el desarrollo. Uno de los más populares es *PrimeFaces*, una librería de componentes visuales *open source* desarrollada y mantenida por Prime Technology [7]. *PrimeFaces* tiene una gran sección de componentes con un diseño sencillo e innovador y de uso sencillo para el

desarrollador que, además, puede personalizar el aspecto o el funcionamiento de dichos componentes.

- **Hypertext Preprocessor (PHP)** [8][9]: es un lenguaje de código abierto muy popular, especialmente adecuado para el desarrollo web, y que puede ser incrustado en HTML. El código es interpretado por un servidor web con un módulo de procesado de PHP que genera la página web resultante. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

- **Active Server Pages (ASP**) [10]: es una tecnología de páginas activas de Microsoft del lado del servidor y ha sido comercializada como un anexo a *Internet Information Services* (IIS). Permite el uso de diferentes scripts y componentes junto con HTML para mostrar páginas generadas dinámicamente. Su principal limitación es que sólo se puede utilizar en Microsoft Windows.

Posibles tecnologías que se pueden aplicar con el uso del lenguaje de Java:

- **Hibernate** [5] es una herramienta que permite mapear los atributos entre una base de datos relacional y el modelo de objetos de una aplicación con el uso anotaciones en los *beans* de las entidades. Esto hace que no sea necesario usar consultas SQL para traer datos y relaciones.

- **Java Persistence API (JPA)** [24] es una especificación para acceder, persistir y gestionar los datos entre los objetos Java y una base de datos. JPA es solo una especificación y no implementa la API, sino que es Hibernate quien se encarga de implementar esta especificación JPA [5].

- **Enterprise JavaBeans** [25] es un componente (*enterprise beans*) que agrupa un conjunto de funcionalidades definidas por el programador y que puede ser reutilizado y ensamblado en distintas partes de una misma aplicación o incluso en distintas aplicaciones.

Independientemente de las tecnologías utilizadas, para ejecutar una aplicación web es necesario disponer de un servidor de aplicaciones. De entre las diferentes alternativas existentes en el mercado, cabe destacar las siguientes:

- **Tomcat** [11] desarrollado por Apache bajo el proyecto Jakarta. A diferencia de otros servidores, éste funciona como un contenedor de servlets. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle Corporation.

- **JBoss (Wildfly)** [12] desarrollado por RedHat. JBoss es un servidor de aplicaciones Java EE. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible una máquina virtual de Java. Es software libre y de código abierto, sujeto a los requisitos de la GNU Lesser General Public License (LGPL), version 2.1.

- **Glassfish y Weblogic** [13] desarrollados por Oracle Corporation. Glassfish es un servidor de aplicaciones de software libre que implementa las tecnologías definidas en la plataforma Java EE 5 y permite ejecutar aplicaciones que siguen esta especificación.

Actualmente Oracle ha dejado de dar soporte a Glassfish y la nueva versión que sustituye a Glassfish es Weblogic.

2.1.3 Almacenamiento de datos

Finalmente, para la tercera capa del patrón MVC, existen una gran cantidad de gestores de bases de datos relacionales entre los que cabe destacar las siguientes:

- **MySQL** es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation [14][15]. Está considerado como uno de los gestores de bases de datos más populares junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web. Entre los productos *open source* es posiblemente el más utilizado. De hecho, de acuerdo al estudio de mercado realizado por la compañía Jelastic en el primer cuatrimestre de 2016 (Figura 2-2), MySql lidera el mercado de bases de datos con un porcentaje de 65%.

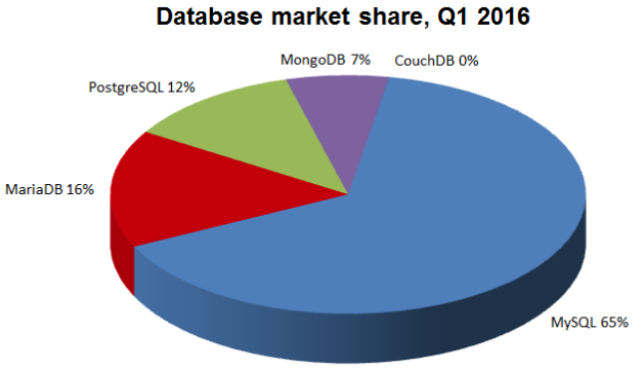


Figura 2-2: Estudio de mercado de bases de datos para el primer trimestre de 2016 realizado por Jelastic [16]

- **PostgreSQL** es un sistema de gestión de bases de datos relacionales orientado a objetos y libre publicado bajo la licencia PostgreSQL [17]. Utiliza un modelo cliente/servidor y emplea multiprocesos para asegurar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

- **Oracle Database (Oracle)** es un sistema de gestión de base de datos de tipo objeto-relacional, comercial y de pago [18]. Se considera a Oracle Database como uno de los sistemas de bases de datos más completos. Ha sido diseñado para controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio reduciendo así los costes y los riesgos de la pérdida de información.

- **Microsoft SQL Server** es un sistema de manejo de bases de datos relacionales desarrollado por la empresa Microsoft [19]. El lenguaje de desarrollo utilizado es Transact-SQL (TSQL), una implementación en estándar ANSI del lenguaje SQL y empleado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL). Solo está disponible para sistemas operativos Windows de Microsoft.

2.2 Environmental Sensor Data Repository

Environmental Sensor Data Repository (ESDR) es un repositorio de datos de código abierto creado por el Instituto de Robótica de la Universidad de Carnegie Mellon [26] con el objetivo de almacenar, recuperar y visualizar de forma rápida y segura datos procedentes de sensores ambientales comúnmente representados como series temporales. Por tanto, ESDR únicamente necesita conocer la marca de tiempo y el valor de lectura para cada uno de los distintos tipos de sensores con los que se desee trabajar [20] [1].

Los datos del sistema ESDR se guardan en un almacén de datos (*datastore*) de código abierto implementado en lenguaje C++ que proporciona inserciones y extracciones de datos extremadamente rápidas, lo que permite realizar visualizaciones rápidas. El sitio web de ESDR (esdr.cmucreatelab.org) proporciona una interfaz API REST al almacén de datos, facilitando la lectura / escritura de datos. Los metadatos se almacenan en una base de datos MySQL.

La mayor ventaja de utilizar el sistema ESDR es la rápida visualización de datos procedentes de sensores ambientales. Permite visualizar datos de forma eficiente para resoluciones temporales que pueden ir desde los segundos hasta años enteros, realizándose el cambio de una resolución a otra casi instantáneamente. Usando Java y una base de datos relacional no se podría llegar a esta rapidez de representación de datos. Esta es la principal razón de mantener ESDR como la herramienta de visualización de datos de sensores.

La Figura 2-3 muestra una imagen del sistema ESDR adaptado por el grupo GNB para el almacenamiento y visualización de los datos recogidos por las narices electrónicas diseñadas e implementadas por el grupo e instaladas en diferentes puntos de la EPS. Se puede ver que se han filtrado los datos entre junio de 2016 y febrero de 2017 así como los sensores de los que se desea mostrar la información.

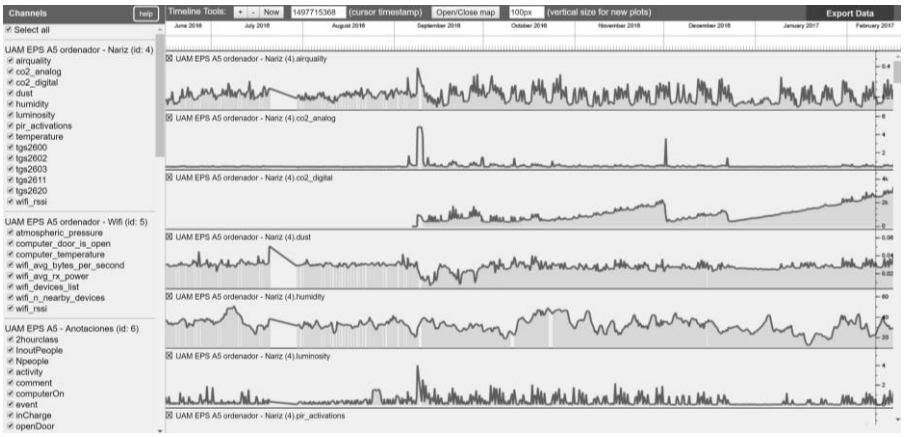


Figura 2-3: Ejemplo del sistema ESDR utilizado para el almacenamiento y visualización de la información procedente de las redes de sensores multimodales instaladas en la EPS.

[1] Nota: En este documento no se detallará la instalación del sistema ESDR, pero se puede encontrar esta información en https://github.com/CMU-CREATE-Lab/esdr.

2.2.1 Conceptos y terminología ESDR

ESDR utiliza su propia terminología para referirse a los diferentes elementos que componen el sistema. En particular, el sistema ESDR tiene clientes, usuarios, productos, dispositivos, *feeds*, canales y tiles [20].

- **Cliente**: ESDR utiliza OAuth2 para la autenticación. OAuth2 es un protocolo de autorización que permite a los usuarios autoricen a terceros a acceder a su información sin que éstos tengan que conocer las credenciales del usuario.

- **Usuario:** persona que se ha registrado con ESDR y puede poseer uno o más productos, dispositivos o *feeds*. Cuando un usuario inicia sesión, lo hace en nombre de un cliente OAuth2.

- **Producto:** tipo de nariz artificial, por ejemplo, la nariz electrónica diseñada e implementada por el Grupo de Neurocomputación Biológica de la UAM.

- **Dispositivo**: es una nariz artificial real y tiene un número de serie único.

- ***Feed***: es una instalación particular de una nariz artificial a la que se le asignan diferentes características como ubicación (latitud y longitud), exposición (interior o exterior), visibilidad (pública o privada), etc. El *feed* estará compuesto por uno o varios canales.

- **Canal**: es un sensor que mide uno o más aspectos de su entorno, tales como temperatura, humedad, conteo de partículas, voltaje de la batería, etc. Por tanto, típicamente la nariz artificial estará compuesta por varios sensores y se considerará que toda la información proporcionada por esa nariz vendrá dada por un *feed* que, a su vez, se compondrá de las señales individuales de cada uno de los sensores que forman la nariz (canales).

- **Tiles**: son ficheros recuperables con extensión JSON de los datos de los distintos canales de un *feed*. Un tile contiene como máximo 512 puntos de datos y está asociado con una marca de tiempo de inicio y duración. La cantidad de tiles devuelto dependerá de la línea de tiempo que el usuario desee. ESDR también tiene soporte para una recopilación multi-tile donde puede obtener datos de varios canales de múltiples *feeds* con una sola petición GET, que permite la visualización de datos de más de un sensor simultáneamente.

En este Trabajo de Fin de Grado se utilizarán los términos usuario, *feed* y canal, ya que por el momento no es necesario diferenciar distintos productos y dispositivos.
Además, se ha detectado que la información proporcionada sobre los productos y dispositivos no afecta potencialmente a la funcionalidad de los *feeds*. Por ejemplo para definir un producto es necesario especificar un nombre, su vendedor, descripción, etc, pero el valor de estos campos no tiene ningún tipo de implicación a la hora de trabajar con los datos del *feed*. Incluso la propia interfaz gráfica del sistema ESDR tampoco hace uso de estas informaciones. En el siguiente capítulo se especificará como interaccionará la aplicación a desarrollar con el sistema ESDR.

2.2.2 Redes de sensores instaladas en la EPS

Actualmente se monitoriza la actividad en el aula 5 y el seminario B-351 de la Escuela Politécnica Superior.

La nariz del aula 5 está compuesto por cinco grupos de sensores:

- **Nariz artificial instalada cerca del ordenador**: básicamente es una agrupación de sensores que miden la calidad del aire, la cantidad de dióxido de carbono, de polvo, de humedad, de luminosidad y de temperatura.

- **Nariz artificial instalada cerca del proyector**: también se tienen los sensores instalados cerca del ordenador más algunos otros, como un acelerómetro, un sensor que mide la temperatura de acelerómetro, un sensor que mide la presión atmosférica, etc.

- **Sensores multimodales cerca del ordenador:** dispositivo compuesto por sensores que miden la intensidad de la señal de wifi, la temperatura del ordenador y presión atmosférica.

- **Sensores que recogen estadísticas de partículas**: sensor comercial (Speck Sensor, https://www.specksensor.com) distribuido por los creadores de ESDR y que mide la cantidad de partículas que hay en el aire, la humedad y la temperatura.

- **"Sensores" que guardan datos de anotaciones**: son canales donde se guardarán los datos recogidos por agentes externos (no sensores), estos datos se obtienen a partir de los formularios rellenados por profesores de la EPS. No son sensores propiamente dichos, pero para la integración de estos datos en ESDR se tratan como tal.

En el seminario B-351 hay dos grupos de sensores:

- **Nariz artificial** formada por sensores que recogen datos de temperatura, número de partículas en suspensión, concentración de dióxido de carbono y humedad.

- **Sensores que recogen estadísticas de la señal de wifi:** dispositivo similar al que se utiliza para recoger estadísticas de wifi del aula 5.

Aunque no es objeto de este trabajo, hay que tener en cuenta que a partir de los datos obtenidos de estos sensores y de las anotaciones realizadas por miembros del grupo de investigación y colaboradores externos se pueden realizar diferentes análisis para determinar la capacidad de las narices artificiales para, por ejemplo, estimar el grado de atención según el tipo de actividad (clases con diapositiva o de pizarra) o el número de personas en el aula. Una funcionalidad que sí se implementará en la aplicación a desarrollar es la generación de matrices con frecuencias de muestreo constantes y etiquetado de patrones que servirán de punto de partida para los estudios del grupo de investigación. En la sección 3.2 se darán más detalles sobre esta funcionalidad.

3 Análisis

Como se ha comentado en el capítulo 1, el propósito de este Trabajo de Fin de Grado es analizar los requisitos, diseñar, implementar y validar un sistema software de gestión datos y generación de matrices para cálculos estadísticos sobre los datos recogidos por la red. En las siguientes secciones se detallarán las funcionalidades de este sistema.

3.1 Análisis de las herramientas a utilizar

Tras el análisis realizado en el capítulo 2, se tiene una visión básica de las nuevas las tecnologías y herramientas a usar para el desarrollo e implementación del sistema. En base a este análisis se han tomado las siguientes decisiones:

- Se utilizará **MySql** como la base de datos del sistema. Además de ser la base de datos más popular en las aplicaciones web, el sistema ESDR al que se tiene que contactar esta nueva aplicación también utiliza esta base de datos para guardar metadatos.
- **Java** será el lenguaje principal de esta aplicación, por lo que se utilizará JSP con apoyo de PrimeFaces para implementar la parte visual.
- Se ha optado a utilizar **Wildfly** como servidor de aplicaciones. No se ha querido usar Tomcat porque es un contenedor de servlets que, aunque implementa Java Servlet y JavaServer Pages, no es suficientemente potente para manipular datos. Wildfly además de incluir casi todas las funcionalidades que tiene Tomcat, proporciona Java Enterprise Edition (JEE), incluyendo Enterprise JavaBeans y muchas otras tecnologías tales como Hibernate, un servicio de persistencia objeto/relaciones y consultas para Java que no necesita de consultas SQL para gestionar los datos de la base de datos.

Se detallará el uso de estas herramientas en el capítulo 5.

3.2 Análisis de requisitos

3.2.1 Análisis de requisitos funcionales

A continuación se especifican todos los requisitos funcionales del sistema:

1. La aplicación permitirá el acceso a distintos tipos de usuarios. Habrá dos tipos de usuarios: los usuarios externos (EXT) que no pertenecen al grupo de investigación, y los usuarios internos (INT) asociados a miembros del grupo de investigación.
   1.1. La aplicación prestará distintas funcionalidades según el tipo de usuario que interactúe con la misma.
   1.2. El acceso se realizará con la introducción de los siguientes campos:
      - Nombre de usuario
      - Contraseña
2. La aplicación permitirá el registro de nuevos usuarios. El formulario de inscripción solicitará los siguientes campos (los campos indicados con * son campos obligatorios):
      - Nombre de usuario*
      - Nombre*

- Primer apellido*
- Segundo apellido
- Contraseña
- Repetición de la contraseña.
- Tipo*: es el tipo de usuario.
- Tipo de actividad: tipo de actividad que suele realizar en el aula/seminario; por ejemplo, si las clases normalmente son expositivas con diapositivas o son clases teóricas que solo se utilizan la pizarra, etc.
- Dispositivo: el tipo de dispositivo suele utilizar: ordenador de sobremesa disponible en el aula, ordenador portátil o ninguno.

3. La aplicación permitirá a los usuarios la modificación de sus datos: nombre de usuario, nombre, primer apellido, segundo apellido, contraseña, tipo de actividad y dispositivo.
4. La aplicación permitirá a los usuarios la eliminación de su cuenta.
5. La aplicación permitirá a los usuarios el cierre de sesión.
6. La aplicación mostrará una vista con todos los usuarios y permitirá ver los detalles de cada usuario.

A continuación, se detallarán los requisitos según el tipo de usuario:

<u>Tipo de usuario EXT</u>:

7. Crear formularios externos: estos formularios sirven para dar información relativa a un evento.
    - Lugar*: lugar donde se ha realizado el evento.
    - Fecha*: fecha en la que tuvo lugar el evento.
    - Hora de inicio*: hora de inicio en la que tuvo lugar el evento.
    - Hora de fin: hora de fin en la que tuvo lugar el evento. Si no se especifica, fecha fin será fecha de inicio más un segundo para representar una anotación puntual en un instante de tiempo.
    - Número de personas*: número de personas que ha asistido al evento
    - ¿Puerta abierta? *: si el evento se ha realizado en un aula o sala, la puerta estaba cerrada o abierta
    - Tipo de actividad*: tipo de actividad que se ha realizado, por ejemplo, clases teóricas utilizando pizarra.
    - Dispositivo*: tipo de dispositivo que se ha usado, ordenador, portátil o ninguno.
    - Comentario / Incidencia
8. Visualizar el detalle de un evento. Un evento es una actividad que se realiza en un aula o sala. Tiene asociada información sobre el instante de inicio y fin del evento, la persona a cargo del mismo, el número de asistentes, etc. Por ejemplo, un evento sería la clase de Sistemas Informáticos I que tuvo lugar el día 3 de noviembre de 2016 entre las 13 y las 14 horas en el aula 5 y a la que asistieron 40 personas y cuyo responsable era Ruth Cobos Pérez.
9. Visualizar todos los eventos que tienen lugar en una determinada aula o sala.

<u>Tipo de usuario INT:</u>

10. Crear formularios externos: será el mismo formulario que para un usuario EXT.

11. Crear formularios internos: formularios similares a los formularios externos (ver requisito 7), pero en los que se añade el campo de persona a cargo puesto que no es necesariamente el propio usuario que rellena el formulario.
12. Validar los formularios creados. Cuando se rellena un formulario, solo se guardarán estos datos en la base de datos de la aplicación web. Es necesario que haya un usuario INT, quien se encargará de validar este formulario para poder enviar estos datos al sistema ESDR. Como de momento solo hay un único grupo de canales de anotaciones (el de aula 5), se guardarán los datos en esos canales.
13. Visualizar todos los formularios del sistema: se podrán visualizar todos los formularios rellenados, tanto los enviados a ESDR como los que no han sido enviados de momento. Para los que ya han sido enviados, se especificarán quién ha valido dichos formularios.
14. Visualizar las series temporales con las lecturas de los sensores (acceso al sistema ESDR). Se podrá acceder al sistema ESDR desde esta aplicación. El sistema ESDR permite visualizar los datos de los distintos sensores instalados. En la adaptación de la plataforma ESDR se ha incluido un campo *timestamp* donde se muestra la marca temporal asociada al instante de tiempo donde se tiene el cursor tal como se muestra en la Figura 3-1. Esta información puede ser utilizada por los usuarios cuando se quiere especificar los instantes inicial y final para el borrado de datos o generación de matrices de datos.

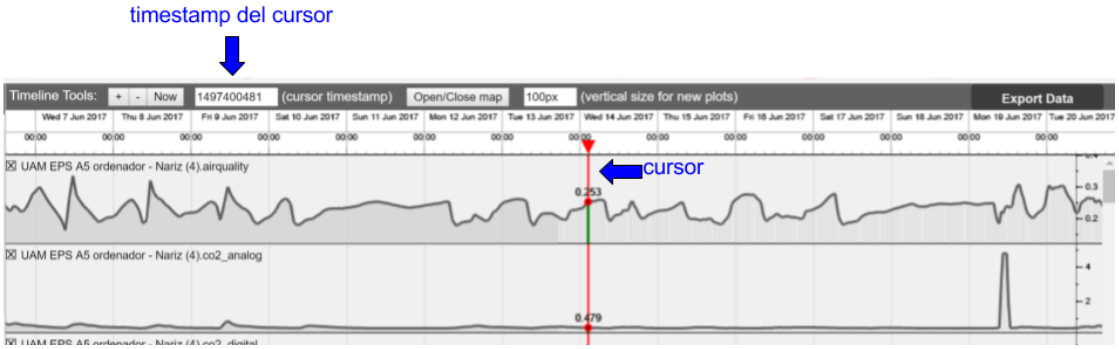


Figura 3-1: Vista ESDR con el cursor

15. Visualizar todos los eventos que tienen lugar en una determinada aula o sala.
16. Crear un nuevo evento. Será necesario rellenar los siguientes campos:
    - Nombre de evento*
    - Fecha de inicio*
    - Fecha de fin*
    - Persona a cargo*
    - Comentario

    Permite crear evento con o sin repeticiones.

    16.1. Evento sin repetición: es un evento que solo se realiza una única vez.

    16.2. Eventos con repetición: es un evento que tiene lugar cada n días o semanas, como por ejemplo las clases de una asignatura durante un determinado cuatrimestre.

    16.2.1. Permitir crear eventos cada n días.

    16.2.2. Permitir crear eventos cada n semanas y poder elegir los días de la semana.

    16.2.3. Finalizar el evento con repetición tras n repeticiones

    16.2.4. Finalizar el evento con repetición en una fecha en concreto.
17. Editar evento: los cambios en los eventos no afectarán a los formularios creados y validados para dicho evento.

18. Borrar evento: solo se permitirá borrar un evento que no tiene un formulario asociado.
19. Visualizar el detalle de un evento: lo mismo que en usuarios EXT.
20. Crear nuevo lugar. El lugar es el aula o la sala donde se instalarán los sensores. ESDR no permite añadir un lugar a un *feed*, solo la latitud y la longitud, por lo que es difícil saber de forma clara donde están instalados. Los lugares de esta aplicación web permitirán agrupar los *feeds* de manera intuitiva. A la hora de validar un formulario, se enviarán los datos al *feed* que está definido por el lugar especificado por dicho formulario. Para dar de alta un nuevo lugar, se deben especificar los siguientes campos:
    - Nombre*
    - Descripción
    - *Feed* de formulario: cuando se valida un formulario, se enviará los datos al *feed* de formulario del lugar especificado en el formulario.
    - *Feed* instalados: una lista de *feeds* instalados. Un *feed* sólo puede estar instalado en un lugar.
21. Visualizar todos los lugares de la aplicación.
22. Editar lugar. Sólo afectará al lugar que se quiere editar. Los formularios ya enviados a ESDR, no se verán afectados.
23. Visualizar el detalle un lugar
24. Visualizar *feeds* públicos: es una consulta a ESDR y solo se mostrarán los *feeds* con la propiedad *isPublic* igual a uno.
25. Crear un nuevo *feed*. Como la propiedad apiKey de un *feed* (un identificador propio y único para cada feed asignado por el sistema ESDR) no es recuperable mediante consultas ESDR, ésta se guardará automáticamente en la base de datos de la aplicación. La apiKey es necesaria para realizar operaciones de escritura, por ejemplo, en el envío de datos de un formulario y el borrado de datos. Para crear un nuevo *feed* se solicitará al usuario la siguiente información:
    - Nombre*
    - Exposición*: campo definido por ESDR. Se admite tres valores: *indoor*, *outdoor*, o *virtual*.
    - Es público: no es un campo obligatorio, pero si no se pone nada, se sobreentiende que no es público.
    - Es movible: no es un campo obligatorio, pero si no se pone nada, se sobreentiende que no es movible.
    - Latitud: se admite valores entre -90 a 90, ambos inclusive.
    - Longitud: se admite valores entre -180 a 180, ambos inclusive.
    - Id del dispositivo*: identificador del dispositivo al cual pertenece el *feed*.
    - Lugar*: lugar donde esté instalado el *feed*. Tener en cuenta que este campo no se guardará en ESDR.

    Cuando se crea un *feed*, se enviará directamente a ESDR y este será el encargado de guardar la mayor parte de información de un *feed*. La aplicación web solo guardará el apiKey y lugar.
26. Editar *feed*: la mayoría de campos de un *feed* no son editables, ya que están guardados en ESDR.
27. Visualizar el detalle de un *feed*: casi todos los campos se obtendrán de ESDR, excepto lugar y apiKey.
28. Borrar datos del sistema ESDR: ya que ESDR no permite eliminar valores para *timestamps* que ya han sido grabados en el sistema, lo que se hará será meter un valor inválido cuyo valor se podrá especificar en las propiedades de configuración. El borrado de datos se podrá realizar en base a diferentes filtros:

28.1. Según fechas y los canales seleccionados por el usuario
28.2. Según *timestamp* y los canales seleccionados por el usuario. El campo *timestamp* se podrá coger del campo cursor *timestamp* de la vista de ESDR.

29. Generar matrices de datos para el posterior análisis de los mismos. Se podrá generar dos tipos de matrices:
    - Matriz con frecuencias de muestreo constante: seleccionar fechas o *timestamps*, el *feed* que se toma como referencia de tiempos, el valor de la frecuencia constante y los canales. El campo *timestamp* se podrá coger del campo cursor *timestamp* de la vista de ESDR.
    - Matriz de datos etiquetados: seleccionar todos los campos para unificar frecuencias y demás definir los intervalos de no eventos (datos de clase negativa o sin asistentes en el aula).
30. Visualizar las propiedades del sistema. Las posibles propiedades de configuración serán:
    - Dirección de la página de ESDR a conectar.
    - Directorio del fichero de autenticación con ESDR
    - Valor inválido utilizado para borrar datos.
    - Directorio donde se guardarán ficheros creados para hacer peticiones a ESDR.
    - Directorio donde se encentrarán los scripts de Python para generar matrices.
    - Valor de interpolación para los formularios.
    - Dirección donde se encontrará desplegado el servidor.
    - Flag para especificar si se quiere que se borren los ficheros creados para hacer peticiones a ESDR.
    - Flag para especificar si se quiere que se borren los ficheros de los matrices creados.
31. Editar una propiedad del sistema: solo se permitirá cambiar el valor de una propiedad.
32. Añadir nuevos tipos de actividades: identificador de tipo de actividades, ya que ESDR no permite guardar texto como dato de los canales de *feeds* (sensores), por lo que en algún sitio se deberá especificar qué valor numérico coinciden con qué actividad.
33. Visualizar todos los tipos de actividades
34. Añadir nuevos tipos de eventos: tiene el mismo objetivo que los tipos de actividades
35. Visualizar todos los tipos de eventos
36. Añadir nuevas personas a cargo: tiene el mismo objetivo que los tipos de actividades
37. Visualizar todas las personas a cargo

3.2.2 Análisis de requisitos no funcionales

1. La interfaz gráfica de la aplicación web deberá ser sencilla e intuitiva, de forma que cualquier usuario sea capaz de explotar de forma completa la funcionalidad de la aplicación.
2. El sistema web se presentará en castellano, ya que de momento el ámbito de uso está acotado a profesores de EPS.
3. La aplicación web deberá ser compatible en distintos navegadores y/o Sistemas Operativos.
4. La aplicación web contará con un sistema de cifrado de datos para proteger las contraseñas de los distintos usuarios en la base de datos.

3.3 Interacción con el sistema ESDR

En algunas partes de la aplicación es necesario interactuar con el sistema ESDR. En esta sección se indica qué módulos se encargan de esta funcionalidad y se describe cómo tiene lugar esta interacción.

Se identificaron cinco grandes módulos donde es necesaria la comunicación con el sistema ESDR tal y como se muestra en la Figura 3-2:

- Validación de formularios: será necesario enviar datos a un *feed*. Al tratarse de una operación de escritura, será imprescindible tener el apiKey de dicho *feed*.
- Eliminación de datos de ESDR: también es una operación de escritura en un *feed*, por lo que se necesitará el apiKey.
- Creación de *feed*: en este caso se necesitará que el usuario este autenticado en el sistema ESDR para poder crear una *feed*.
- Generación de matrices: no es ninguna operación de escritura, solo son consultas, por lo que no se requerirá ninguna autenticación.
- Cualquier consulta de *feeds*: son operaciones de lectura, por lo que no se requerirá ninguna autenticación. Entre estas operaciones se encuentran la visualización de detalle de un *feed*, el acceso a ESDR, etc.

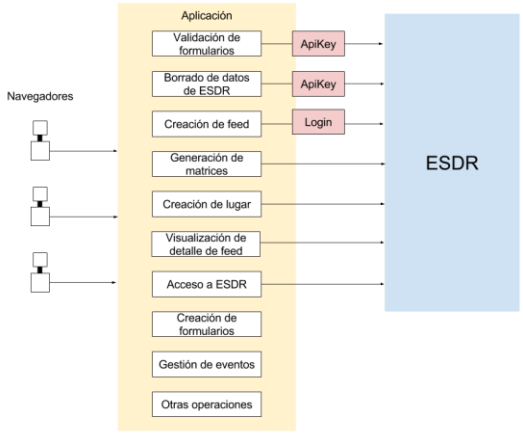


Figura 3-2: Esquema de comunicación con ESDR

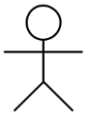
Para la comunicación con ESDR se han utilizado comandos *curl* contra la API REST que ESDR proporciona. En el anexo C Interacción con ESDR con comandos curl se presentan algunas consultas para recuperar datos de ESDR. En Java existen varias formas de usar el comando *curl* similares a las opciones que ofrece una consola Linux:

- Mediante la clase *URL*, pero esta opción no permite pasar parámetros.
- Mediante la clase *HttpURLConnection* [22] que es capaz de conectar con un servidor HTTP. Esta opción fue descartada al no parecer muy sencilla e intuitiva.
- Mediante la clase *ProcessBuilder* [23] que permite ejecutar los comandos de la consola en código Java y también permite añadir argumentos. Para ejecutar un comando basta con crear una instancia de *ProcessBuilder* y pasarle como argumentos el comando y todos los parámetros que se quieran añadir y después hacer un *start()* para ejecutarlo. Además es posible obtener los mensajes de salida y de error. Debido a su uso sencillo y flexibilidad, esta fue la alternativa escogida para hacer las consultas de ESDR.

4 Diseño

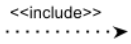
4.1 Diagrama casos de uso

Para mostrar de una forma más clara las interacciones entre usuarios y la aplicación, se han creado diagramas de casos de uso según el catálogo de requisitos funcionales del capítulo 3.



Actor: es un rol de usuario interacciona con el sistema.

Tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación.



La relación de inclusión sirve para enriquecer un caso de uso con otro y compartir una funcionalidad común entre varios casos de uso.

Una operación/tarea específica que se realiza tras una orden de algún agente externo.

La Figura 4-1 se muestra los escenarios en los que la aplicación interactúa con un usuario que no pertenece al grupo de investigación (usuario tipo EXT). El usuario solo tendrá permisos para ver eventos, ver detalle de evento y rellenar formulario externo y para realizar estas acciones será necesario haberse registrado en el sistema previamente.

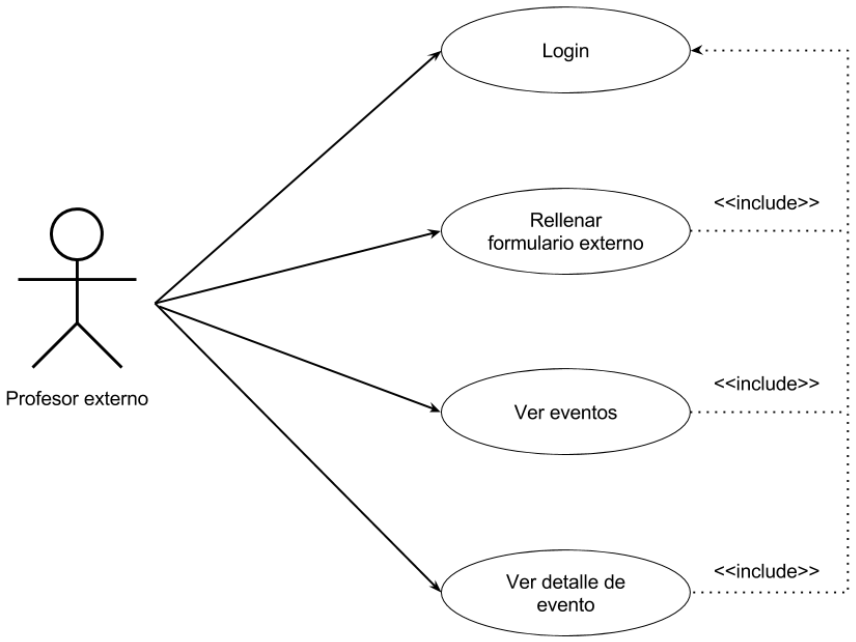


Figura 4-1: Diagrama de casos de uso de un usuario que no pertenece al grupo de investigación

Las Figuras Figura 4-2 Figura 4-7 muestran diagramas de las acciones que podrá realizar un usuario que pertenece al grupo de investigación (usuario tipo INT).

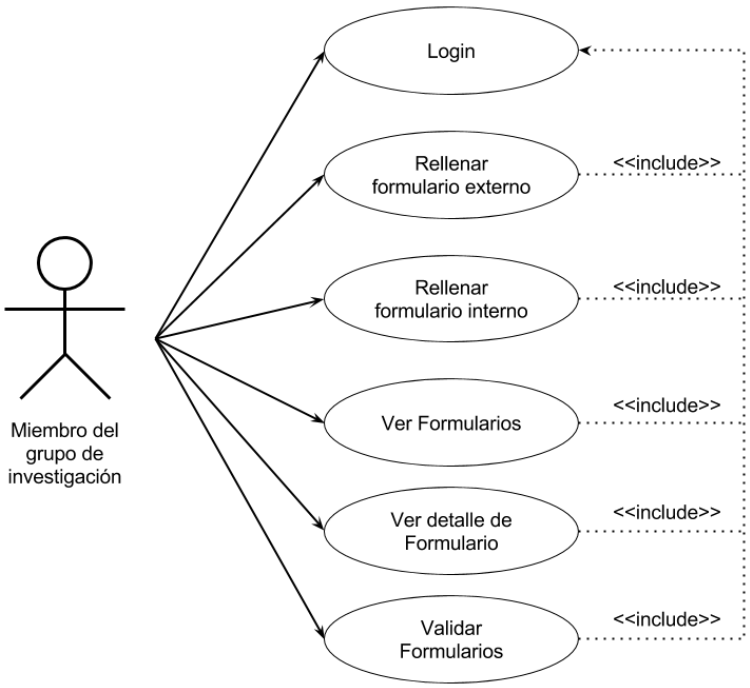


Figura 4-2: Diagrama de casos de uso de un usuario que pertenece al grupo de investigación y las acciones que puede realizar con los formularios

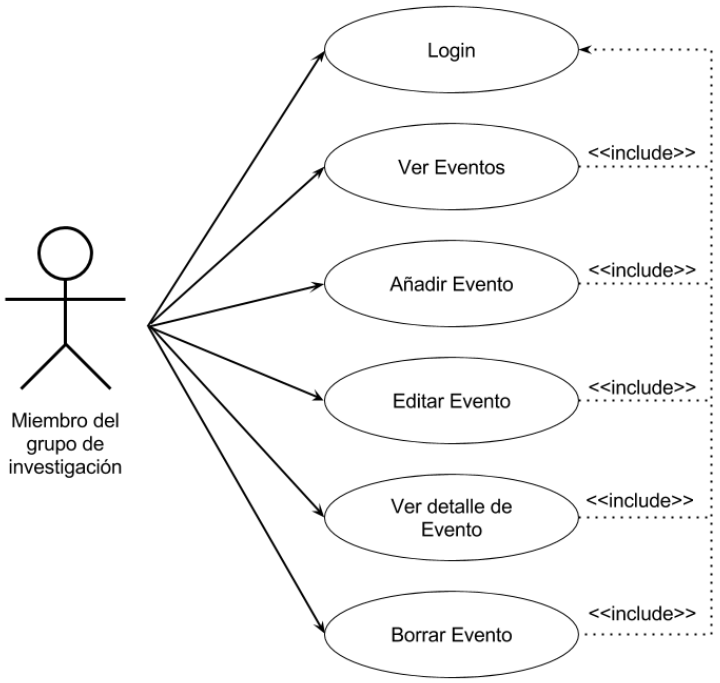


Figura 4-3: Diagrama de casos de uso de un usuario que pertenece al grupo de investigación y las acciones que puede realizar con los eventos

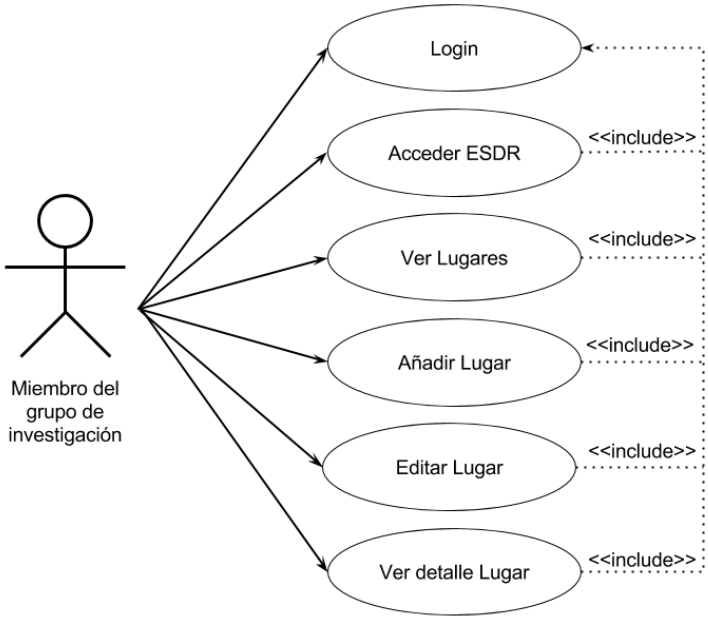


Figura 4-4: Diagrama de casos de uso de un usuario que pertenece al grupo de investigación y las acciones que puede realizar con los lugares y acceder ESDR

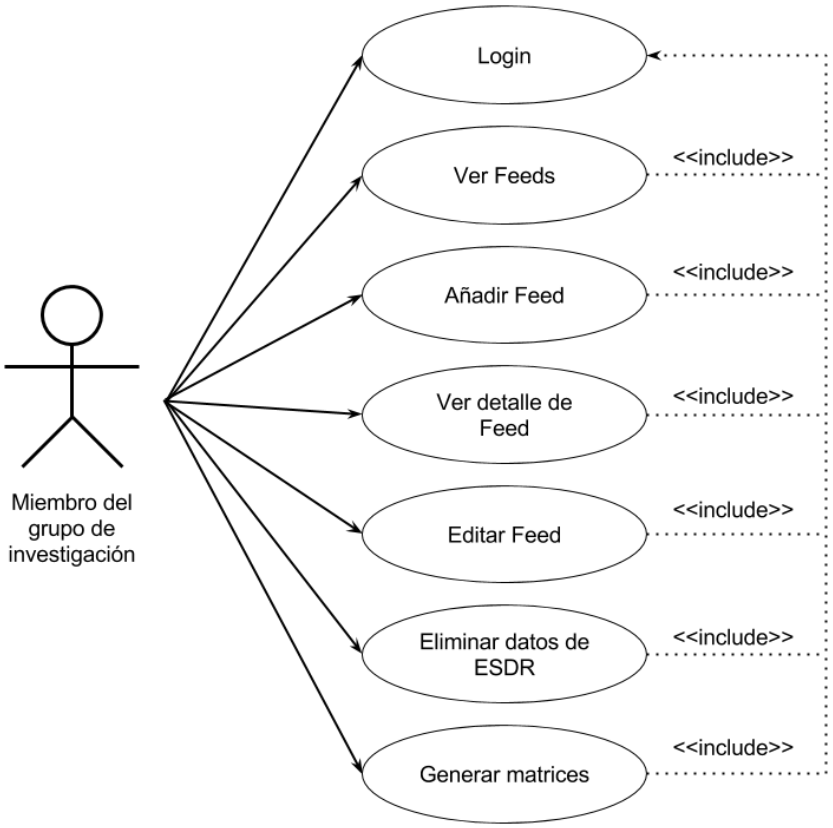


Figura 4-5: Diagrama de casos de uso de un usuario que pertenece al grupo de investigación y las acciones que puede realizar con los *feed*, eliminar datos de ESDR y generar matrices

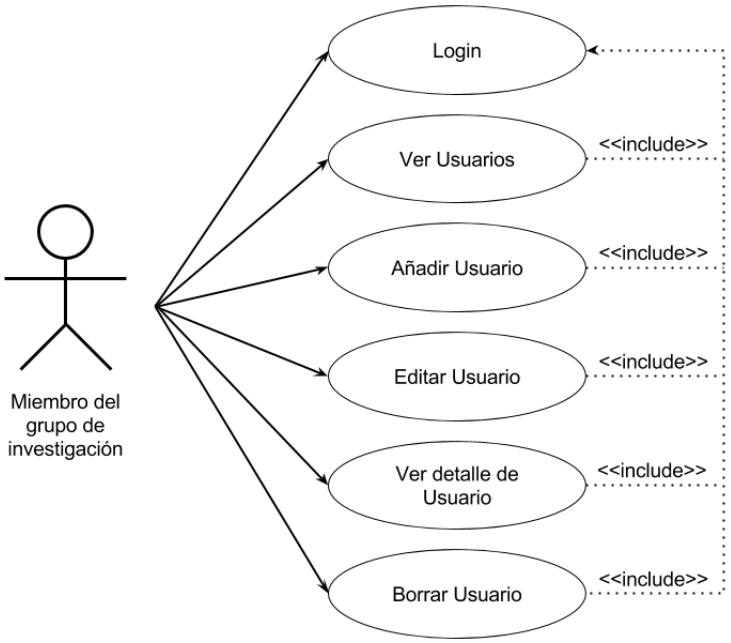


Figura 4-6: Diagrama de casos de uso de un usuario que pertenece al grupo de investigación y las acciones que puede realizar con los usuarios

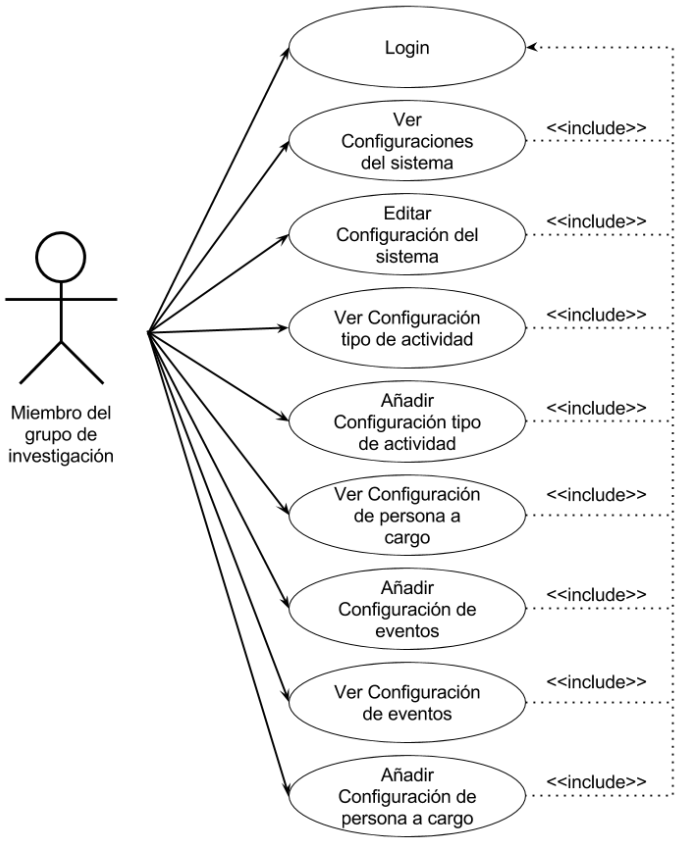


Figura 4-7: Diagrama de casos de uso de un usuario que pertenece al grupo de investigación y las acciones que puede realizar con las configuraciones del sistema y configuraciones de tipo de actividades, eventos y persona a cargo

4.2 Diagrama entidad relación

Para representar las entidades de la base de datos de la aplicación y las relaciones entre ellas se ha hecho uso de un diagrama entidad-relación (Figura 4-8).

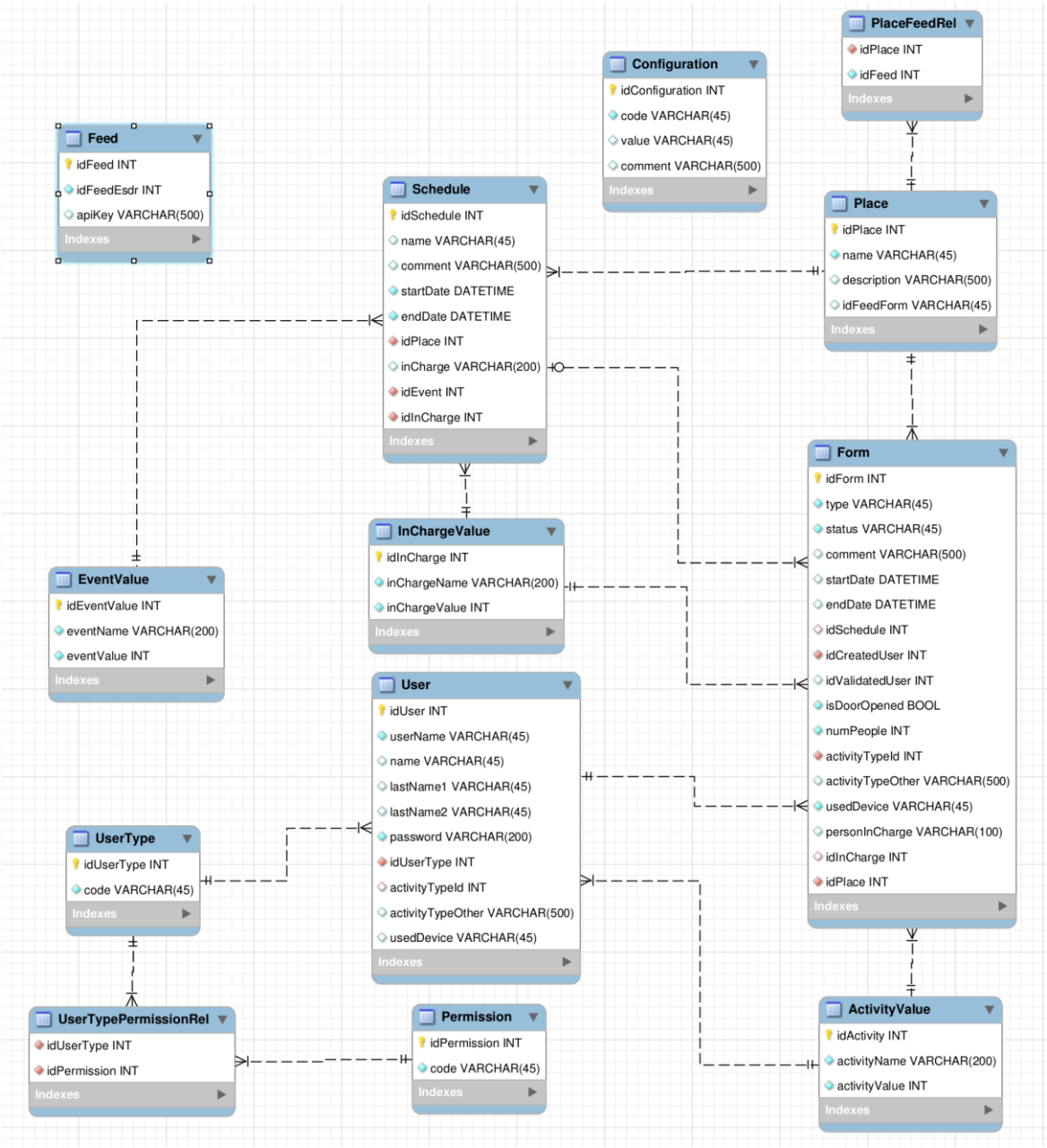


Figura 4-8: Diagrama entidad relación

Dado que el sistema ESDR se encarga de guardar todos los datos relativos a la lectura de los sensores, la aplicación no se encargará de almacenar esta información. Sí será responsabilidad de la aplicación almacenar la información capturada a través de los formularios a rellenar por los usuarios y que incluyen datos como el número de personas en un evento o el tipo de actividad. En particular, para esta aplicación se han creado las siguientes tablas:

1. Información de **usuarios (UserDataType)**: se almacenarán toda información relativa a los usuarios como nombre, contraseña, etc.

2. Información de **eventos (ScheduleDataType)**: se guardarán toda información relativa a los horarios o eventos como por ejemplo fecha de inicio, fecha de fin, lugar, etc.
3. Información de **formularios (FormDataType)**: se guardarán toda la información relativa a los formularios tanto externos como internos, como fecha de inicio, fecha de fin, lugar, evento, etc.
4. Alguna información relevante y no recuperable de los ***feeds*** (sensores) **(FeedDataType)**, como por ejemplo el lugar donde está instalado el sensor o el apiKey del *feed* que permite las operaciones de escritura.
5. Los **permisos de los usuarios (PermissionDataType)**: definirá todos los permisos del sistema, que facilitará el control de las vistas.
6. Los **lugares** (aulas o salas) **(PlaceDataType)**: se guardarán información de aula o sala donde se instalarán sensores.
7. Algunas **configuraciones para el sistema ESDR**, como identificador de tipo de actividades, eventos, etc, ya que ESDR no permite guardar texto como dato de los canales de *feeds* (sensores), por lo que en algún sitio se deberá especificar qué valor numérico coinciden con qué actividad u otras informaciones.
    a. **Valor de tipo de actividad (ActivityValueType)**: esta tabla servirá para relacionar un valor numérico con un tipo de actividad.
    b. **Valor de evento (EventValueType)**: tiene el mismo objetivo que ActivityValueType, pero con los eventos.
    c. **Valor de persona a cargo (InChargeValueType)**: tiene el mismo objetivo que ActivityValueType, pero con las personas a cargo.
8. Las **configuraciones de la aplicación web (ConfigurationDataType)**: se usará para guardar configuraciones del sistema.
9. **Relación de lugar con *feed* (PlaceFeedRelType).**
10. **Tipo de usuario (UserTypeDataType):** tabla con información sobre los tipos de usuarios y sus permisos.

4.3 Estructura de las vistas

Para que todas las vistas de la aplicación tengan un aspecto uniforme, se ha decidido que todas seguirán un mismo formato. Tendrían (a) una cabecera donde llevará el nombre de la aplicación, (b) una barra de menú superior, que permitirá al usuario navegar por todas las vistas y, por último, (c) el contenido propio de cada vista tal y como se muestra en la Figura 4-9.

Figura 4-9: Estructura simplificada de las vistas

Para el diseño de la parte de interfaz web de la aplicación, se han creado maquetas de las posibles vistas. Estas maquetas ayudarán a definir el aspecto de cada vista, dónde se implementarán cada una de las funcionalidades y cómo se relacionarán las vistas entre sí. Todas las maquetas se encuentran en el Anexo B Maquetas.

Se tendrán principalmente tres tipos de acciones (a) acciones relacionadas con los formularios, (b) acciones para gestionar los datos y (c) acciones relacionadas con la configuración del sistema. Para realizar cualquier acción será necesario iniciar sesión. Una vez se haya iniciado sesión, se mostrará la barra de menú según el tipo de usuario, ya que no todos los usuarios tendrán los mismos permisos.

5 Desarrollo

En este capítulo se describen los detalles de desarrollo de la aplicación web. Como se ha comentado en los capítulos anteriores, se utilizará Java como lenguaje de programación y para tener un mejor proceso de desarrollo se seguirá el patrón MVC. No se hará uso del lenguaje SQL para la manipulación de datos y se utilizará Hibernate. Se empleará Eclipse como herramienta de programación (edición y estructuración de código) y Maven como herramienta de compilación, ya que Maven tiene un amplio repositorio de librerías y dependencias, por lo que es muy fácil incluirlas y utilizarlas en el proyecto.

5.1 Patrón Modelo Vista Controlador (MVC)

El patrón Modelo Vista Controlador (MVC) cuyas funcionalidades son:

- **Modelo**: tiene las clases que se encargan de acceder a la información y también para actualizar su estado.
- **Vista**: contiene el código que va a producir la visualización de las interfaces de usuario, es decir, el código que permitirá *renderizar* los estados de la aplicación en HTML.
- **Controlador**: contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, una búsqueda de información, etc.

5.1.1 Modelo: Hibernate, Java Persistence API y Enterprise JavaBeans

En el desarrollo del modelo se han utilizado Hibernate, Java Persistence API y Enterprise JavaBeans. Estas herramientas han facilitado los procesos de manipulación de datos, no siendo necesario el uso de otros lenguajes como SQL.

En la Figura 5-1 se puede ver el flujo de datos desde que la información es requerida por la capa de negocio utilizando los EJB hasta que ésta es recuperada/almacenada en la base de datos a través de la interfaz de consultas JPA implementada por Hibernate

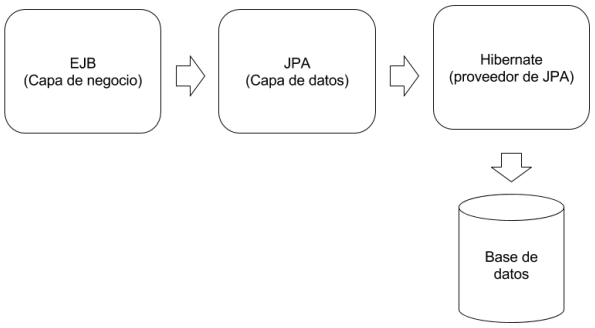


Figura 5-1: Diagrama EJB, JPA, Hibernate

El uso de estas herramientas ofrece una serie de ventajas como son (a) la posibilidad de trabajar con la base de datos por medio de entidades en vez de consultas SQL; (b) la capacidad de diseñar un paradigma cien por cien orientado a objetos (coincide con Java); (c) la eliminación de errores en tiempo de ejecución; y (d) la mejora en el mantenimiento

del software, ya que si algún día se quiere cambiar de base de datos sólo será necesario de cambiar las configuraciones.

Como se puede ver en la Figura 5-2, se ha creado una interfaz de repositorio genérica (*Repository*), donde se declaran todos los métodos comunes a implementar, como por ejemplo crear, actualizar, borrar y buscar por identificador sobre los distintos tipos de entidades: *feeds*, usuarios, eventos, etc. También se ha creado una clase de repositorio que implementa esta interfaz y es donde se tienen los métodos comunes implementados utilizando la tecnología JPA (*AbstractJpaRepository*). Todas las demás interfaces (*EntityRepository*) y clases de DAO (*EntityJpaRepository*) extienden *Repository* y *AbstractJpaRepository* y declaran e implementan sus propios métodos de repositorio. Por ejemplo, la funcionalidad de recuperar eventos entre dos fechas es propia del repositorio de eventos y, por tanto, no se incluye en la interfaz genérica.

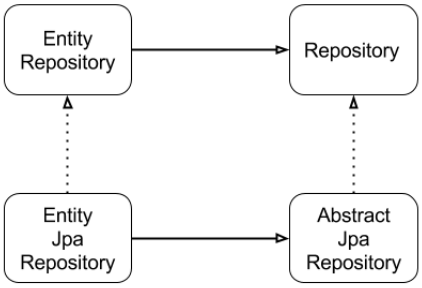


Figura 5-2: Diagrama de clases de DAO simplificado

Las interfaces tienen la función de definir los métodos implementados en las clases de JPA y tienen la ventaja de ocultar al cliente de la clase los detalles de la implementación dado que éste únicamente conoce el prototipo de las funciones de la interfaz. De esta forma, si algún día se quiere cambiar la implementación de las funciones definidas en la interfaz, no será necesario cambiar las clases que usan estos métodos del lado del cliente. Además tener una clase JPA genérica permite reutilizar código, ya que sólo será necesario implementar los métodos no comunes en cada clase. La Figura 5-3 muestra la definición de la interfaz Repository.

```
public interface Repository<T> {

	T create(T t);
	T getById(Integer id);
	T update(T t);
	void delete(T t);
	void delete(List<T> ts);
}
```

Figura 5-3: Interfaz *Repository*

JPA proporciona la interfaz *EntityManager* que permite hacer operaciones con los datos guardados en una base de datos. Estas operaciones son casi las mismas que proporciona SQL; por ejemplo mientras que en SQL una consulta del tipo *insert* permite insertar datos en la base de datos, en JPA es el método *persist* el que tiene la misma funcionalidad. Como se ha dicho anteriormente, JPA solo es una especificación y es Hibernate quien implementa todos los métodos. Para que Hibernate pueda conectarse con la base de datos es necesario indicar el nombre de la unidad de persistencia, que representa un conjunto de entidades que

están mapeadas a la base de datos de la aplicación, en este caso se corresponde con el valor sgdesdr.mysl. Esto se define en el fichero *persistence.xml*, que permite configurar la base de datos que se quiere usar en la aplicación. En la Figura 5-4 se muestra la implementación de la clase *AbstractJpaRepository*. Obsérvese como las implementaciones de los métodos son muy simples ya que simplemente invocan al método correspondiente de la interfaz *EntityManager*.

```
@TransactionAttribute(TransactionAttributeType.SUPPORTS)
public abstract class AbstractJpaRepository <T>
implements Repository<T>{

    @PersistenceContext(unitName="sgdesdr.mysql")
    protected EntityManager entityManager;

    public abstract Class<T> getEntityClass();

    public T create(T t) {
        this.entityManager.persist(t);
        return t;
    }

    public T getById(Integer id) {
        return this.entityManager.find(getEntityClass(),
id);
    }

    public T update(T t) {
        return this.entityManager.merge(t);
    }

    public void delete(T t) {
        t = this.entityManager.merge(t);
        this.entityManager.remove(t);
    }

    public void delete(List<T> ts) {
        for( T t : ts){
            t = this.entityManager.merge(t);
            this.entityManager.remove(t);
        }
    }
}
```

Figura 5-4: Clase *AbstractJpaRepository*

La mayor ventaja de usar JPA es no tener que hace consultas SQL para recuperar datos de la base de datos. Aunque SQL es un lenguaje relativamente sencillo, a veces resulta difícil de mantener cuando se trabaja con consultas complejas. Las clases *CriteriaBuilder* y *Predicate* de JPA permiten la creación de consultas en código Java que son altamente interpretables y fácilmente configurables ya que es relativamente sencillo añadir y quitar parámetros de la consulta. Como ejemplo, la Figura 5-5 muestra una consulta *select* de la tabla de lugares (*PlaceDataType*) que contiene información sobre las ubicaciones en las que se encuentran instaladas las narices artificiales. Este método se puede utilizar sin

ningún tipo de filtro, pero también permite filtrar lugares tanto por identificador como por nombre.

```
public List<PlaceDataType> get(Integer id, String name) {
    CriteriaBuilder builder = entityManager.getCriteriaBuilder();

    CriteriaQuery<PlaceDataType> criteria = builder.createQuery(PlaceDataType.class);
    Root<PlaceDataType> root = criteria.from(PlaceDataType.class);
    criteria.select(root);

    Predicate p = null;

    if(id != null){
        p = builder.equal(root.get(PlaceDataType_.idPlace), id);
    }

    if(name != null){
        if(p == null){
            p = builder.equal(root.get(PlaceDataType_.name), name);
        }else{
            p = builder.and(p,builder.equal(root.get(PlaceDataType_.name), name));
        }
    }

    List<PlaceDataType> results;
    if(p == null){
        results = entityManager.createQuery(criteria.where()).getResultList();
    }else{
        results = entityManager.createQuery(criteria.where(p)).getResultList();
    }

    for(PlaceDataType pl : results){
        Hibernate.initialize(pl.getFeedIds());
        pl.setLoadFeedIds(pl.getFeedIds());
    }
    return results;

}
```

Inicializar instancias para poder hacer la consulta

Si id no es null, significa que se quiere filtrar por id. Entonces añadir id en la consulta

Filtrar por name, si name no es nulo.

Ejecutar consulta sin filtros

Ejecutar consulta con filtros (id y/o name)

En esta caso hay una relación lazy de manyToOne. Hibernate.initialize sirve para recuperar relaciones lazy

Figura 5-5: Clase *PlaceDataJpaRepository*

Todas las clases de repositorio tienen añadida la etiqueta *@Stateless*, esto significa que el *bean* trata cada petición como una transacción independiente que no tiene relación con cualquier solicitud anterior. Por lo que el servidor no retiene información de la sesión.

Se han utilizado servicios web para el acceso a estos métodos de repositorio. Los controladores deben pasar por estos servicios para tener acceso a los datos. En la Figura 5-6 se muestra la definición del servicio web de lugar (*Place*), que tiene tres métodos de servicio:

- **get**: método para recuperar lugares aplicando si se desea filtros de identificador o nombre.
- **store**: método para guardar o actualizar un lugar.
- **getPlaceByFeedId**: método que dado un identificador de un *feed* cualquiera, devuelve el lugar donde está instalado el *feed*.

```
public interface PlaceDataService {

    @WebMethod
    public PlaceDataGetResponse get(PlaceDataGetRequest request);

    @WebMethod
    public PlaceDataStoreResponse store(PlaceDataStoreRequest request) throws InvalidRequestException;

    @WebMethod
    public PlaceDataGetPlaceByFeedIdResponse getPlaceByFeedId(PlaceDataGetPlaceByFeedIdRequest request);
}
```

Figura 5-6: Definición del servicio web de PlaceDataService

5.1.2 Vista: XHTML

A la hora de crear las vistas, además de utilizar la librería de *PrimeFaces* como se comentó en la sección 2.1.2, también se ha hecho uso de otras librerías:

1. **Librería HTML** (http://java.sun.com/jsf/html): permite el uso de componentes básicos de HTML, como *form* para crear formularios, *outputText* para mostrar textos, *outputLink* para crear link, etc.
2. **Librería Core** (http://java.sun.com/jsf/core): define las funcionalidades de etiquetas que sirven para la manipulación y control de datos. Admite el uso de etiquetas como *facet* para definir cabeceras de tabla, *selectItem* para especificar el valor y el identificador cuando hay un menú de selección, o *setPropertyActionListener* para pasar un valor de un atributo a otro, entre otras.
3. **Librería *PrimeFaces*** (http://primefaces.org/ui): permite el uso de los componentes implementados y definidos por *PrimeFaces*, como el componente de calendario o la barra de menú.

Como se ha comentado en el capítulo 4, las vistas están formadas por una cabecera, una barra de menú superior y el contenido propio de cada vista. Con el objetivo de no repetir código de la cabecera y de la barra de menú en cada vista, se han creado *composite view* (agrupaciones de componentes simples) de la cabecera y de la barra de menú y, por tanto, solo es necesario importar estos componentes cada vez que quieran ser utilizados.

5.1.3 Controlador: Managed Bean

En la implementación del controlador, se han utilizado básicamente tres tipos de clases:

- **Clases de controladores**: se encargan de generar la vista e interaccionar con el usuario.
- **Modelos de datos:** sirven para conectar con los servicios web para la obtención, el almacenamiento, la actualización y el borrado de datos.
- **Modelos *lazy*:** utilizados por las tablas o el calendario de la vista para la carga automática de datos en estos componentes.

Sin entrar en los detalles de la implementación, conviene destacar la creación de las clases *BaseController* y *FacesMessageUtils*. La primera implementa los métodos comunes para la interacción con el usuario como comprobar si el usuario actual se ha iniciado la sesión si el usuario tiene permiso para ver una vista determinada, entre otros. Por otra parte, la clase *FacesMessageUtils* se encarga del envío de mensajes informativos desde las clases controladoras a la vista. Consta de un parámetro *level* que indica el tipo de mensaje: error, aviso o mensaje informativo.

5.1.4 Mensajes de las vistas

Todos los mensajes y textos de las vistas de la aplicación se identifican por un código. Existe un fichero, *messages_es.properties* donde se define qué texto en español tiene asociado cada código. Esto facilita la rápida agregación de un nuevo idioma en la aplicación, así como la reutilización de mensajes y textos. Las clases Java necesitan hacer uso de estos mensajes para informar al usuario del resultado (éxito/error) de una operación. Para ello, se ha creado una clase *Message* que se encarga de recuperar el mensaje de texto

asociado a un código de mensaje de acuerdo a las definiciones del fichero *messages_es.properties*.

5.2 Interacción con ESDR

Algunas funcionalidades exigen el acceso al sistema ESDR tales como el envío de formularios con información sobre las anotaciones sobre eventos, la obtención de los valores de un *feed*, etc. Para ello, se ha hecho uso de la clase *ProcessBuilder* para ejecutar los comandos *curl* en código Java, así como de la librería de Jackson para pasear las peticiones a ESDR y repuestas de ESDR en formato JSON [21]. Jackson permite un rápido mapeo de los ficheros JSON a objetos Java definiendo previamente una clase con la misma estructura del fichero. La Figura 5-7 se muestra un ejemplo del uso de la librería Jackson para la interacción con ESDR.

```
String feedQueryUrl = configurationDataRepository.get(CommonConstants.CONFIGURATION_ESDR_URL_CODE).get(0)
        .getValue()+CommonConstants.FEED_QUERY_URL;

ProcessBuilder p=new ProcessBuilder("curl","-X", "PUT", "-H",CommonConstants.JSON_CONTENT_TYPE,
        feedQueryUrl+"/"+feedApikey.getApiKey(), "-d", "@"+filename);
Process shell = p.start();
InputStream errorStream= shell.getErrorStream();
logger.info(CommonConstants.ESDR_CONSOLE_INFO+IOUtils.toString(errorStream, CommonConstants.UFT_8));
InputStream shellIn = shell.getInputStream();
String message = IOUtils.toString(shellIn, CommonConstants.UFT_8);


ObjectMapper mapper = new ObjectMapper();
mapper.configure(SerializationFeature.FAIL_ON_EMPTY_BEANS, false);
mapper.configure(DeserializationFeature.ACCEPT_SINGLE_VALUE_AS_ARRAY, true);

// Convert JSON string to Object
EsdrQueryResponseType result = mapper.readValue(message, EsdrQueryResponseType.class);

if(!(result.getCode().equals(CommonConstants.SUCCESS_CODE_200) && result.getStatus().equals(CommonConstants.SUCCESS_STATUS))){
    deleteFile(filename);
    logger.error(CommonConstants.ESDR_RESPONSE+message);
    throw new BadReponseException("FormDataServiceImpl.store: invalid esdr response", null, result.getCode(),
            result.getStatus(), result.getMessage());
}else{
    logger.info(CommonConstants.ESDR_RESPONSE+message);
}
```

Figura 5-7: Interacción con ESDR

5.3 Generación de matrices de datos

Una de las funcionalidades más importantes de la aplicación es generar matrices de datos para el posterior análisis estadístico y aplicación de algoritmos de aprendizaje automático sobre los datos procedentes de la red de sensores. Para la generación de estas matrices se han utilizado los scripts de Python creados por los profesores del grupo de investigación. Estos scripts devuelven como salida ficheros con campos separados por comas (extensión CSV) a nivel diario y con toda la información asociada a los *feeds* de una determinada ubicación. Además, permiten obtener dos tipos de matrices: matriz con frecuencia unificada y matriz de datos etiquetada.

1. **Matriz de frecuencia unificada**. Como se ha comentado anteriormente, cada uno de los *feeds* de la red de sensores no provee datos a la misma frecuencia. Incluso un mismo *feed* puede ocasionalmente variar sus frecuencias de muestre si, por ejemplo, se ha producido algún tipo de problema con la comunicación wifi. Con el propósito de facilitar el análisis de datos, se genera datos con frecuencias de muestreo uniformes para todos los *feeds* de forma que cada una de las filas de la matriz resultante tiene asociado un *timestamp* e incluye la información de todos los *feeds* asociados a una misma ubicación. Para definir qué *timestamps* se usarán para la uniformización, el

usuario ha de especificar qué *feed* cuyos *timestamps* servirán de referencia y se procederá al cálculo de los nuevos *timestamps* unificados de forma que se minimice la distancia a los *timestamps* de referencia. Para la generación de matrices con frecuencia unificada, primero se deben obtener todos los datos de los *feeds* seleccionados entre las fechas introducidas por el usuario. Esto se hace a través de un script de Ptyhon que realiza las correspondientes llamadas a la API REST de ESDR. Una vez obtenidos estos datos, se ejecuta otro script para calcular una matriz para cada día con frecuencia unificada respecto del *feed* escogido como referencia de tiempo[2]. Después de esto, la aplicación se encargar de escoger y guardar los canales seleccionados por el usuario. Una vez realizadas estas operaciones para todos los días seleccionados, se crea un fichero zip y se envía al controlador.

2. **Matriz de datos etiquetada**. Para la generación de matrices de datos etiquetadas son necesarias las matrices con frecuencias unificadas. La aplicación se encarga de crear una nueva matriz en la que la última columna representa la etiqueta/clase/variable objetivo a considerar por los algoritmos de aprendizaje automático. Para las investigaciones actuales, los miembros del grupo de investigación requerían que en la última columna se incluya el número de gente en el aula. Para ello, es necesario proporcionar a los scripts Python un fichero en el que se indique los periodos de tiempo a considerar como no eventos; es decir, ejemplos de la clase negativa donde no habrá personas en el aula (por ejemplo, domingos de 13 a 14h). Esta información es proporcionada por el usuario desde la interfaz y la aplicación generará el fichero esperado por los scripts de Python. Por otra parte, el usuario también podrá especificar qué canales desea incluir en la matriz y, nuevamente, se creará un fichero con dicha información. Con estos dos ficheros, se ejecutará primeramente el script de Python encargado de generar matrices de datos etiquetadas con información de todos los canales para que después la aplicación desarrollada seleccione los datos correspondientes al conjunto de canales indicados por el usuario. Cada uno de los ficheros diarios resultantes se añadirán al fichero zip que el usuario podrá descargar y que incluirá todas las matrices generadas.

Como puede observarse, los scripts Python requieren de ciertos parámetros que serán solicitados al usuario a través de la interfaz: fecha de inicio y fin de los datos que se desean analizar, *feeds* que se desean tomar como sistema de referencia de tiempos, frecuencia de muestreo a unificar, definición de los periodos de tiempo en los que se considera que no hay ningún tipo de evento en la clase o sala (por ejemplo, domingos de 12 a 13h.), etc. El usuario tiene la opción de elegir si se quiere o no obtener la matriz de datos etiquetados, ya que para generar esto se tiene que especificar más opciones en la interfaz.

Para implementar esta funcionalidad se ha hecho el uso nuevamente de la clase *ProcessBuilder* que se encarga de ejecutar el comando *python*. Antes de ejecutar los comandos para generar las matrices hay que tener en cuenta que el servidor debe tener instalados los paquetes de Python Pandas, tzlocal, matplotlib y Python-tk, puesto que los scripts utilizan estas librerías. Para crear un fichero CSV con las columnas seleccionadas por el usuario a partir de los ficheros CSV generados por los scripts de Python, se ha creado la clase *CSVUtils*, que se encarga de leer y escribir contenidos en un fichero CSV.

[2] La obtención de los datos a nivel diario facilita la generación incremental de matrices y la gestión de ficheros de poco tamaño.

En el Anexo A Diagramas de clases puede encontrar más información sobre las clases y relaciones de las clases en general.

6 Pruebas y resultados

En este apartado se especificarán las pruebas realizadas a lo largo del proceso de desarrollo de la aplicación. Las pruebas se han hecho mientras se iban implementado las funcionalidades, por lo que no se ha llevado a cabo una fase específica de pruebas.

6.1 Pruebas unitarias

Se han creado pruebas unitarias para los métodos que tienen mayor complejidad como por ejemplo el cálculo de las repeticiones de los eventos. Un usuario puede crear un evento que se repite cada n días o cada n semanas especificando los días de la semana. También configurar eventos que finalicen después de n repeticiones o en una fecha en concreto. Este proceso depende varios parámetros de entrada y puede resultar bastante complejo, por lo que se definieron varios casos de test con distintas casuísticas para comprobar su correcto funcionamiento.

Se ha utilizado la librería jUnit para la implementación de estas pruebas que se ejecutan siempre que se compila con el comando *mvn clean install*.

6.2 Pruebas de rendimiento de la generación de matrices

La funcionalidad con mayor coste computacional es la generación de matrices. Por defecto y salvo anomalías en el sensor y/o la comunicación, los sensores que conforman las narices artificiales recogen datos cada cinco segundos, por lo que la generación de matrices de datos para periodos de varios días o meses implica el manejo de una gran cantidad de información.

La generación de matrices con la información de todos los canales se hace con los scripts de Python proporcionados por el grupo de investigación y no tiene un gran impacto en el rendimiento de la aplicación. Sin embargo, la selección del subconjunto de canales especificados por el usuario no estaba implementada en los scripts de Python y se hace en código Java. Durante el transcurso del TFG, no se encontró una manera eficiente de leer y escribir ficheros CSV por columnas, por lo que fue necesario recorrer todas las columnas y filas para guardar solo las columnas que interesan.

1. En una primera versión, se guardaba en memoria la matriz generada por los scripts de Python con la información de todos los canales, para seguidamente seleccionar el conjunto de columnas deseadas y devolver esta submatriz al controlador. El controlador era el responsable de crear el fichero final. Se guardaban todos los datos en memoria porque esta era la forma más rápida de realizar el proceso. Esta estrategia funciona correctamente si se quiere generar matrices para pocos días (entre uno y dos días). Para un mayor número de días, el rendimiento empezaba a decaer considerablemente, llegando incluso a obtenerse casos de error del tipo *"java.lang.OutOfMemoryError: Java heap space"* por no tener suficiente memoria.

2. En la segunda versión de la implementación, se iban escribiendo los datos en otro fichero CSV a medida que el fichero con la información con todos los canales se leía. En este caso, el controlador solo se encarga de pasar el fichero al usuario. Se ha

probado a generar matrices para pocos días y no se ha notado diferencia respecto a la primera versión. También se han hecho pruebas para muchos un mayor número de días (hasta quince días) en las que se ha comprobado la ausencia de errores de memoria y se han obtenido tiempos de ejecución inferiores a dos minutos.

6.3 Producto final

A continuación se muestran algunas pantallas de la aplicación web desarrollada.

Como se ha dicho anteriormente, la aplicación cuenta con un sistema de Login (Figura 6-1) que requiere del nombre de usuario y contraseña para iniciar la sesión.

Figura 6-1: Vista de la pantalla de Login

Existen dos tipos de usuarios, usuarios externos al grupo de investigación (usuarios EXT) y usuarios del propio grupo de investigación (usuario INT).

Primeramente se presentarán las vistas para un usuario tipo EXT. El **usuario EXT** solo tiene permiso para acceder a los formularios propios de su tipo de usuario (formulario externo) y a la página de gestión de eventos. La vista del formulario externo (Figura 6-2) permite el usuario rellenar los datos relativos a un evento. Los campos "Tipo de actividad" y "Dispositivo" se rellenan automáticamente si están predefinidos en su perfil. Por ejemplo, un profesor que indique en su registro que suele dar clases expositivas con diapositivas y que utiliza su propio ordenador o portátil no necesita completar esta información en el formulario externo salvo que haya variaciones sobre este comportamiento "por defecto".

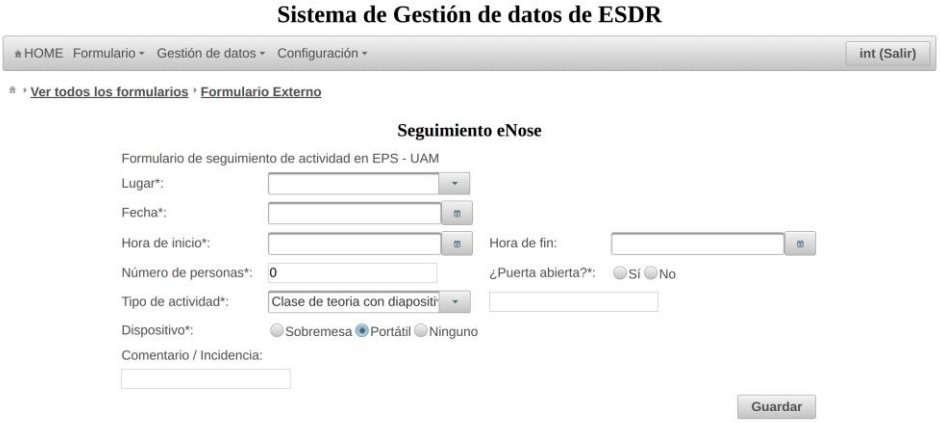


Figura 6-2: Vista de Formulario externo

La vista de Gestión de Eventos permite al usuario visualizar todos los eventos y poder anotar un evento desde la vista de detalle de un evento (Figura 6-3); es decir, rellenar un formulario externo para el evento seleccionado. Conviene destacar que cuando el usuario selecciona un evento del calendario y pulsa el botón “anotar”, el usuario es redirigido a la vista de Formulario externo en el que se auto rellenan campos adicionales a los definidos en su perfil, como el lugar, la fecha y la hora (Figura 6-4).

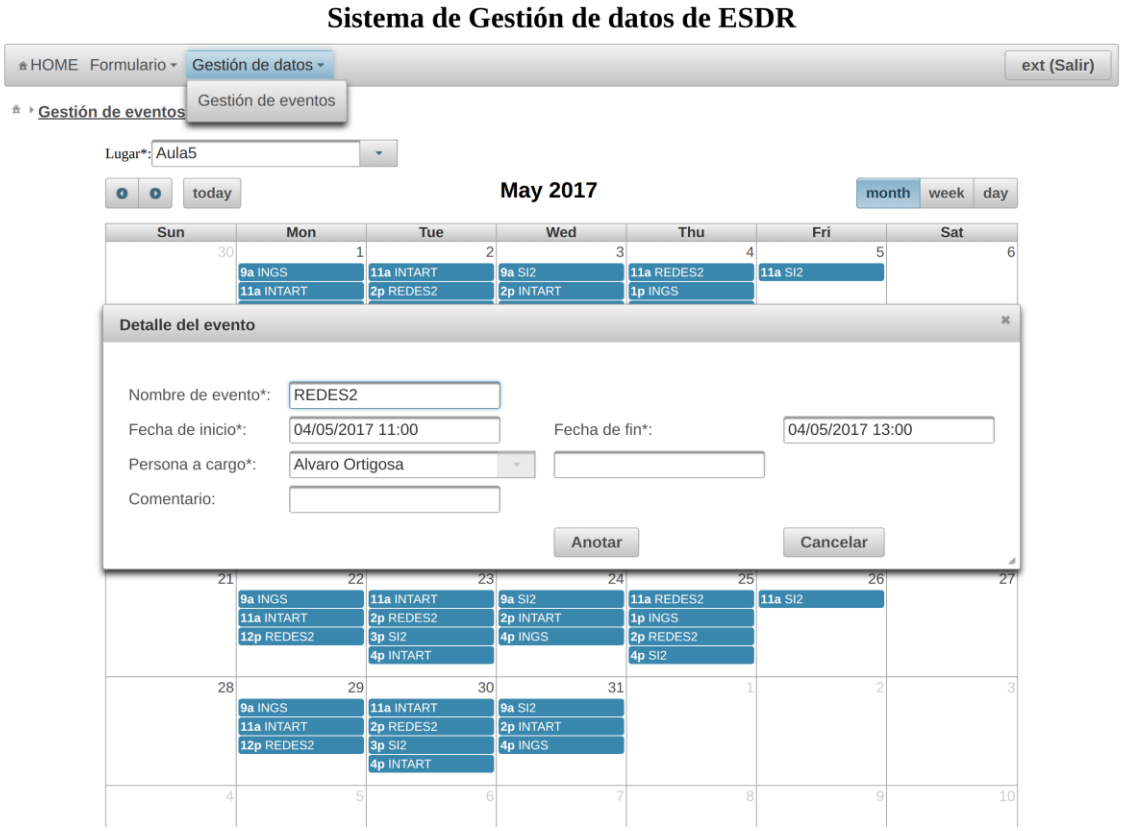


Figura 6-3: Vista de Detalle de evento para usuario EXT

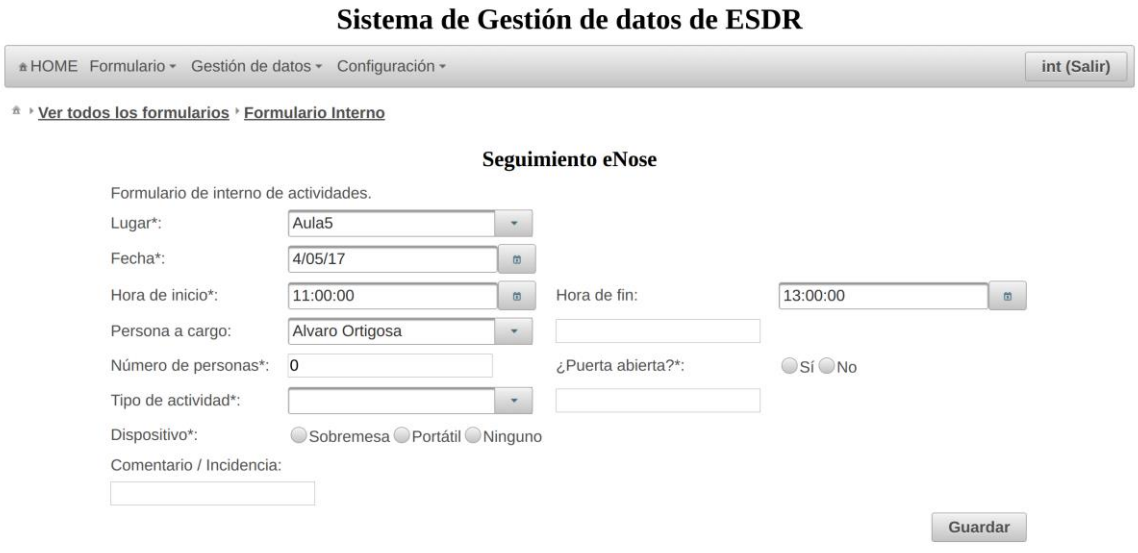


Figura 6-4: Vista de Formulario externo redireccionado desde anotar evento

Un usuario INT, además de tener los permisos de un usuario EXT, puede realizar otras operaciones como validar formularios. Una vez validados, la información contenida en los formularios se envía al sistema ESDR de forma transparente al usuario. En la Figura 6-5 se muestran tres formularios, el primero ya está validado por el usuario INT, el segundo está seleccionado por el usuario y el tercero está sin seleccionar y sin validar.

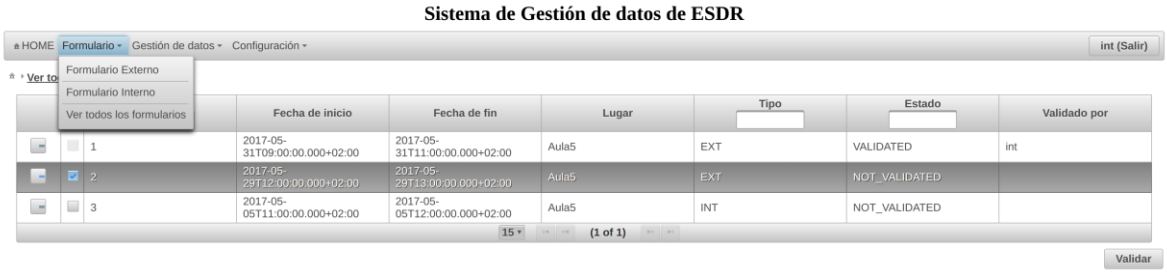


Figura 6-5: Vista de Ver todos los formularios

Un usuario INT también puede crear eventos con o sin repetición (Figura 6-6), borrarlos, editarlos y anotar formularios internos (Figura 6-7). A la hora de crear un evento, las fechas se auto rellenarán según el día seleccionado en el calendario. La posibilidad de definir eventos recurrentes en el calendario ha sido de especial relevancia puesto que de esta forma los eventos correspondientes a clases programadas durante todo un cuatrimestre solo tienen que ser definidos una vez por uno de los miembros del grupo de investigación.

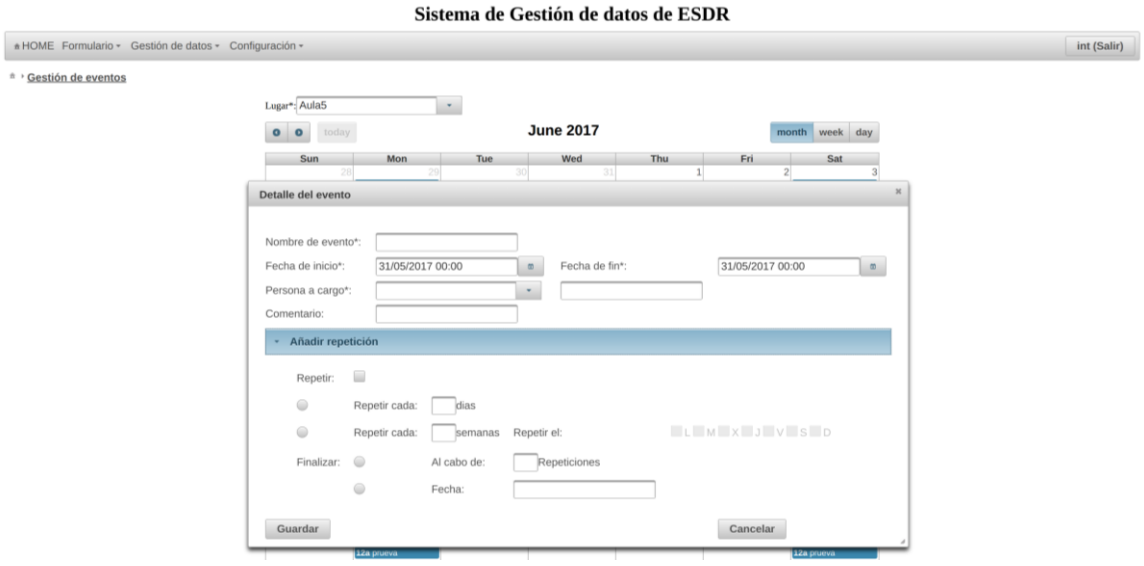


Figura 6-6: Vista de Crear evento

Figura 6-7: Vista de Detalle de evento para usuario INT

Cuando se anota un formulario interno, no se auto rellenan los campos “Tipo de actividad” y “Dispositivo” como pasaba con el formulario externo (Figura 6-8) debido a que cuando se quiere rellenar un formulario interno, es muy posible que se está rellenado en nombre de alguna persona a cargo que no tiene usuario en esta aplicación.

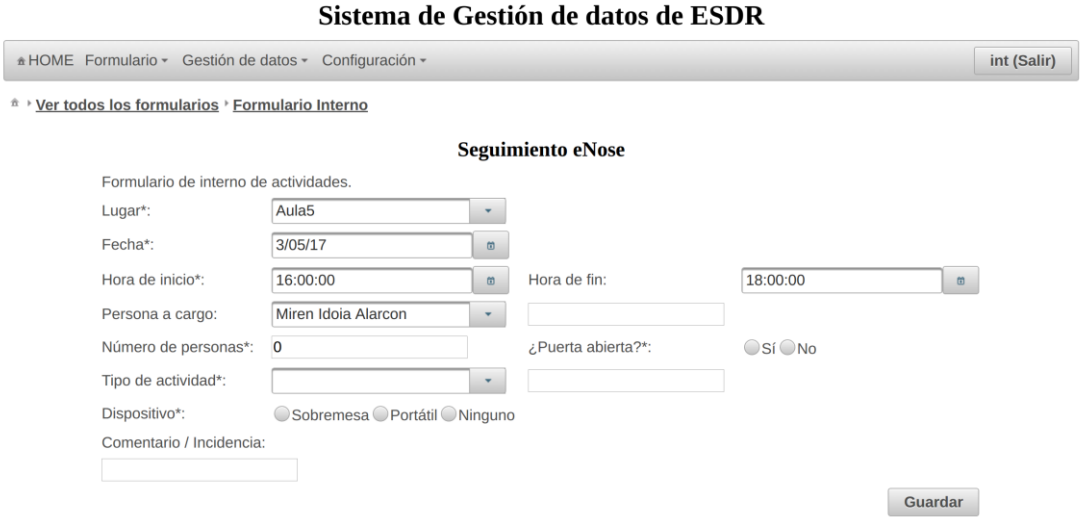


Figura 6-8: Vista de Formulario interno

Entre otras funcionalidades, la información detallada de un *feed* también es accesible por un usuario INT. En la Figura 6-9 muestra el detalle de un *feed* llamado aula_5. En esta vista se pueden visualizar informaciones tanto sobre el *feed* (nombre, exposición, si es público, si es movible, latitud, etc.) como las informaciones de los canales que los que se compone el *feed* (la fecha de inserción del primer dato, valor mínimo, etc). La mayoría de

información que presenta esta vista es obtenida de ESDR, excepto los campos Lugar y apiKey. Para más información sobre estos dos campos, consúltese la sección 3.2.1.

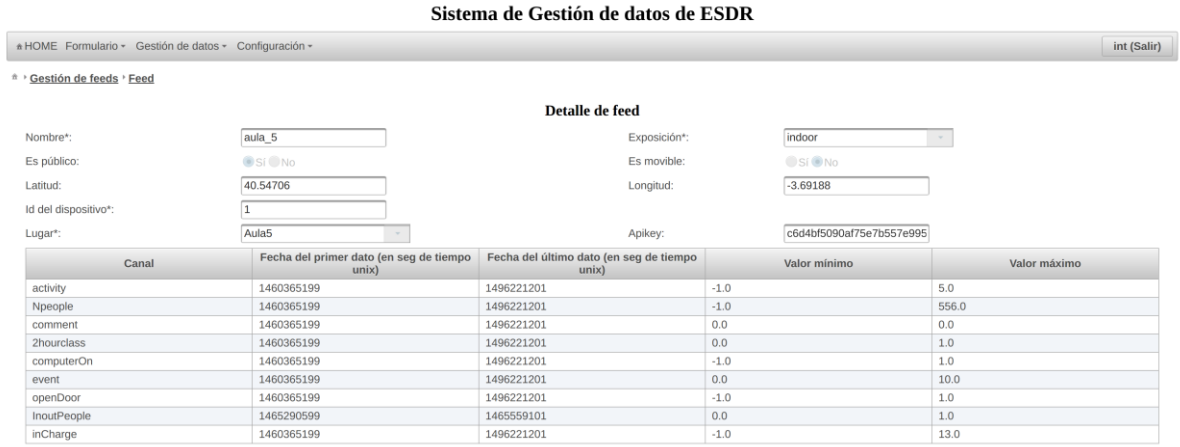


Figura 6-9: Vista de detalle de *feed*

A continuación se muestra en la Figura 6-10 una de las vistas más importantes de la aplicación, la interfaz para construir matrices de datos. Para generar esta vista, es necesario hacer consulta a ESDR para obtener de forma dinámica información de los *feeds* para que el usuario elija qué canales quiere incluir en la matriz de datos a generar. Tras la selección de fechas de inicio y fin de los datos o *timestamps* de inicio y fin así como de los canales a considerar, se produce la descarga de un fichero zip que contiene todos los ficheros de las matrices generadas (un fichero por día).

Figura 6-10: Vista de Generación de matrices

En la Figura 6-11 se presenta la vista de gestión de propiedades del sistema. Se puede editar el valor de cada propiedad. Cuando esta información se almacena correctamente en la base de datos, el controlador envía un mensaje informativo al usuario.

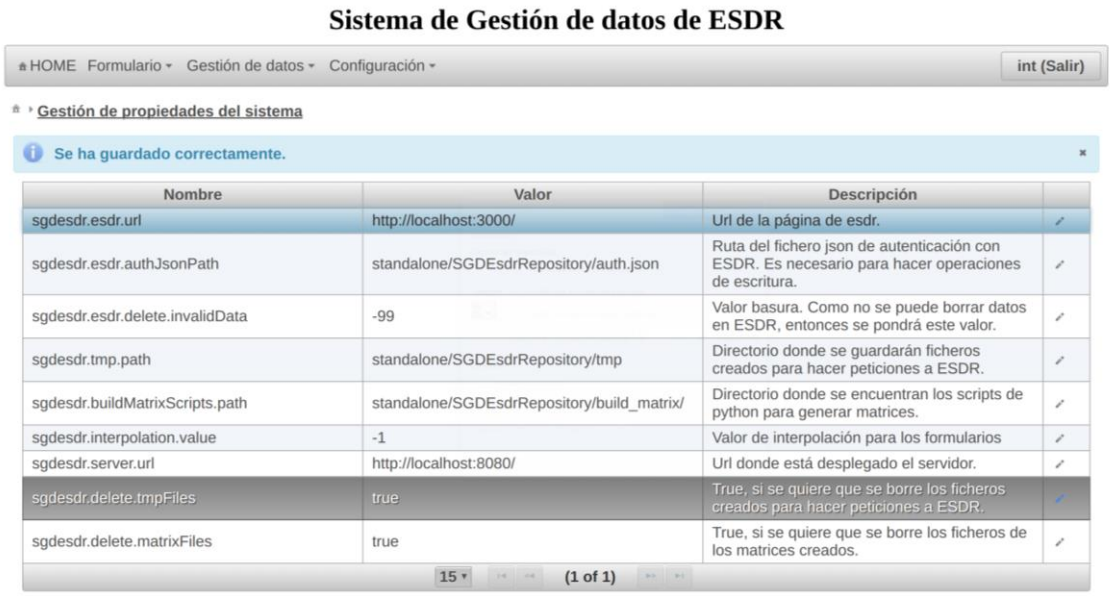


Figura 6-11: Vista de gestión de propiedades del sistema

Por último, se presenta la vista de creación de usuario (Figura 6-12). Aquí es donde se configuran los valores predefinidos de "Tipo de actividad" y "Dispositivo". Si el usuario no rellena los campos obligatorios, la aplicación lo detectará y mostrará un mensaje de error informativo. También conviene destacar que por defecto se precarga un usuario *Admin* de tipo INT especificado el fichero *dml.sql*. Después de instalar la aplicación es necesario usar este usuario para crear los nuevos usuarios que se necesiten.

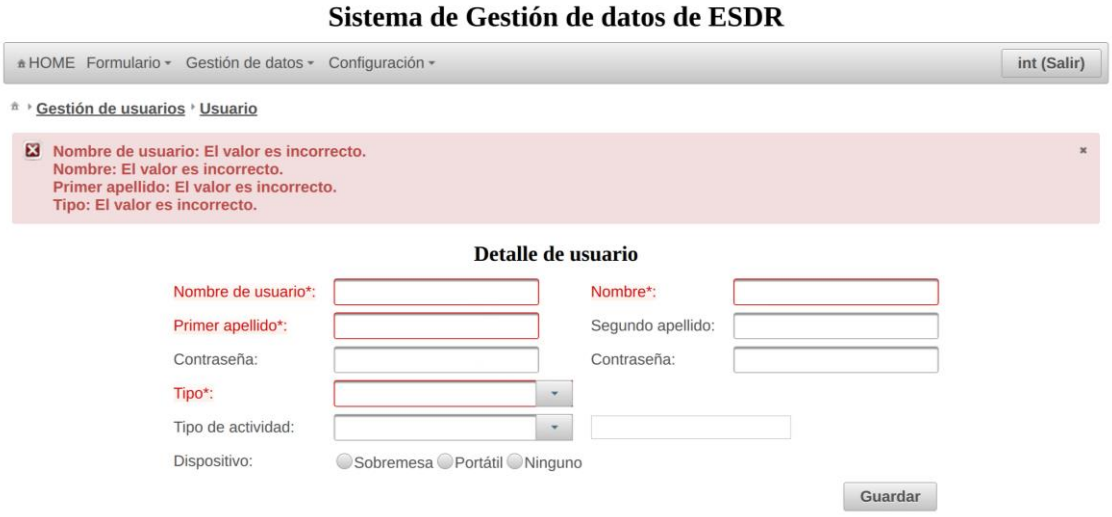


Figura 6-12: Vista de creación de usuario

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El objetivo de este Trabajo Fin de Grado es el desarrollo de una aplicación web que facilite la comunicación con el sistema ESDR para el almacenamiento y visualización de datos recogidos por redes de sensores multimodales instaladas en distintos puntos de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid. Los resultados obtenidos han sido satisfactorios, puesto que se han cumplido los requisitos inicialmente planteados en términos de (i) gestión de eventos: creación de eventos con y sin repetición y anotación de los formularios; (ii) gestión de datos del sistema ESDR: envío de los datos de los formularios de anotación y borrado de los datos en el sistema ESDR, (iii) gestión de usuarios: existencia de usuarios con distintos permisos, y (iv) generación de matrices de datos para el análisis de los mismos. El sistema desarrollado aliviará la carga de trabajo que supone transferir datos de anotaciones a ESDR y la generación de matrices mediante un pipeline de scripts que se venía ejecutando de forma manual.

Además, la aplicación desarrollada tiene una interfaz sencilla, intuitiva, usable por todo tipo de usuarios y accesible desde cualquier dispositivo. Estas características facilitarán la coordinación interna del grupo de investigación para la gestión de ESDR, ya que todos los miembros tendrán acceso a la aplicación y a ESDR.

La estructuración del código es modular, distinguiendo el cliente del servidor y estructurando cada módulo en paquetes, lo que facilitará la incorporación de nuevas funcionalidades y el mantenimiento.

Finalmente destacar el alto grado de aprendizaje que ha supuesto la realización de este TFG al utilizarse tecnologías actuales y distintas a las que se estudian a lo largo del Grado en Ingeniería Informática como son *JPA* y la librería de *PrimeFaces*.

7.2 Trabajo futuro

La aplicación desarrollada en este TFG cubre las necesidades básicas detectadas por los miembros del grupo de investigación GNB. No obstante, durante la realización del proyecto se han detectado posibles mejoras de cara al futuro:

- Incluir un sistema de Login más seguro utilizando otras *frameworks* como *Apache Shiro* o *Spring Security*.
- Implementar mejoras en términos de rendimiento separando cliente y el servidor en dos máquinas, teniendo cada parte su propio servidor de aplicaciones. En este caso se podría usar *Tomcat* como “servidor” del cliente.
- Integrar en la aplicación toda la información proporcionada por ESDR a nivel de productos y dispositivos dado que la versión actual de la aplicación trabaja a nivel de *feed*.
- Desarrollar una versión para dispositivos móviles.
- Ampliar el alcance la aplicación añadiendo más idiomas, como el inglés.

Referencias

[1] Carlos Sánchez González, "Aplicaciones en capas", SourceForge, Septiembre de 2004, http://oness.sourceforge.net/proyecto/html/ch03s02.html

[2] Pedro J. Casanova Peláez, "Curso de HTML", Universidad de Jaén, http://www.uv.es/siuv/cas/zinternet/web/htmlcurso/htmlcurso.htm

[3] "Hoja de estilos en cascada", Wikipedia, https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

[4] "JavaScript", Wikipedia, https://es.wikipedia.org/wiki/JavaScript

[5] "Hibernate ORM", Hibernate, http://hibernate.org/orm/

[6] "JavaServer Pages", Wikipedia, https://es.wikipedia.org/wiki/JavaServer_Pages

[7] Enrique Viñé Lerma, "Introducción a Primefaces", adictosaltrabajo, 30 de junio de 2010, https://www.adictosaltrabajo.com/tutoriales/introduccion-primefaces/

[8] ¿Qué es PHP?, PHP, http://php.net/manual/es/intro-whatis.php

[9] PHP, Wikipedia, https://es.wikipedia.org/wiki/PHP

[10] Christian Van Der Henst S, "¿Qué es el ASP?", maestrosdelweb, 23 de abril de 2001, http://www.maestrosdelweb.com/aspintro/

[11] "Apache Tomcat", Tomcat, http://tomcat.apache.org/

[12] Hector García Galán , "WildFly, el servidor de aplicaciones Java que multiplica su rendimiento en Cloud", arsys, 8 de marzo de 2017, https://blog.arsys.es/wildfly-cloud/

[13] Bannysolano, "¿Qué es Glassfish?", Banny's WebBlog, 23 de agosto de 2009, https://bannysolano.wordpress.com/2009/08/23/%C2%BFque-es-glassfish/

[14] "Oracle mysql", Oracle, https://www.oracle.com/lad/mysql/index.html

[15] "MySql", Wikipedia, https://es.wikipedia.org/wiki/MySQL

[16] Alex Anikin, "Software Stacks Market Share: First Quarter of 2016", linkedin, 6 de junio de 2016, https://www.linkedin.com/pulse/software-stacks-market-share-first-quarter-2016-alex-anikin

[17] Rafael Martinez Guerrero, "postgreSql-es", Emc2Net, 2 de octubre de 2010, http://www.postgresql.org.es/sobre_postgresql

[18] "Oracle" , *iessanvicente,* https://iessanvicente.com/colaboraciones/oracle.pdf

[19] Albino Quiroz Cercado, "Base de datos en microsoft sql server", slideshare, 26 de junio de 2016, https://es.slideshare.net/AlbinoQuirozCercado/base-dedatosenmicrosoftsqlserver-63450353

[20] Chris Bartley, "Environmental Sensor Data Repository (ESDR)", github, https://github.com/CMU-CREATE-Lab/esdr

[21] Tatu Saloranta, "Jackson Project", github, https://github.com/FasterXML/jackson

[22] "Class HttpURLConnection", oracle, http://docs.oracle.com/javase/8/docs/api/java/net/HttpURLConnection.html

[23] " Class ProcessBuilder", oracle, https://docs.oracle.com/javase/8/docs/api/java/lang/ProcessBuilder.html

[24] Kevin Bowersox, "What is JPA? How does it differ from Hibernate?", tothought, 17 de mayo de 2014, http://blog-tothought.rhcloud.com//post/2

[25] "Introducción a la tecnología EJB", Experto java, http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm

[26] Byron Spice, "Carnegie Mellon Spinoff Introduces Speck, A Personal, Wi-Fi-Connected Air Quality Monitor", Carnegie Mellon University, 16 de marzo de 2015, https://www.cmu.edu/news/stories/archives/2015/march/speck-air-quality-monitor.html

Glosario

ANSI	American National Standards Institute
API	Application Programming Interface
ASP	Active Server Pages
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DAO	Data Access Object
DDL	Data Definition Language
DML	Data Manipulation Language
EJB	Enterprise JavaBeans
ESDR	Environmental Sensor Data Repository
GNU GPL	GNU General Public License
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IIS	Internet Information Services
JEE	Java Enterprise Edition
JPA	Java Persistence API
JSON	JavaScript Object Notation
JSP	JavaServer Pages
LGPL	Lesser General Public License
MVC	Model View Controller
OAuth2	Open Authorization 2
PHP	Hypertext Preprocessor
REST	Representational State Transfer
SQL	Structured Query Language
TSQL	Transact-SQL
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language

Anexos

A. Diagramas de clases

Se han creado diagramas de clases para presentar la estructura de la aplicación mostrando las clases con sus atributos y sus métodos, y las relaciones entre las clases.

En primer lugar, se muestra un diagrama de clases de ***types*** (Figura A-1), que son todos los objetos que podrán ser guardados en la base de datos. Por lo que estas clases coinciden con las tablas de la base de datos y los atributos de las clases también se corresponden con las columnas de las tablas. Para simplificar el diagrama, solo se muestran los atributos de las clases, ya que los métodos que implementan son solo constructores y métodos get y set.

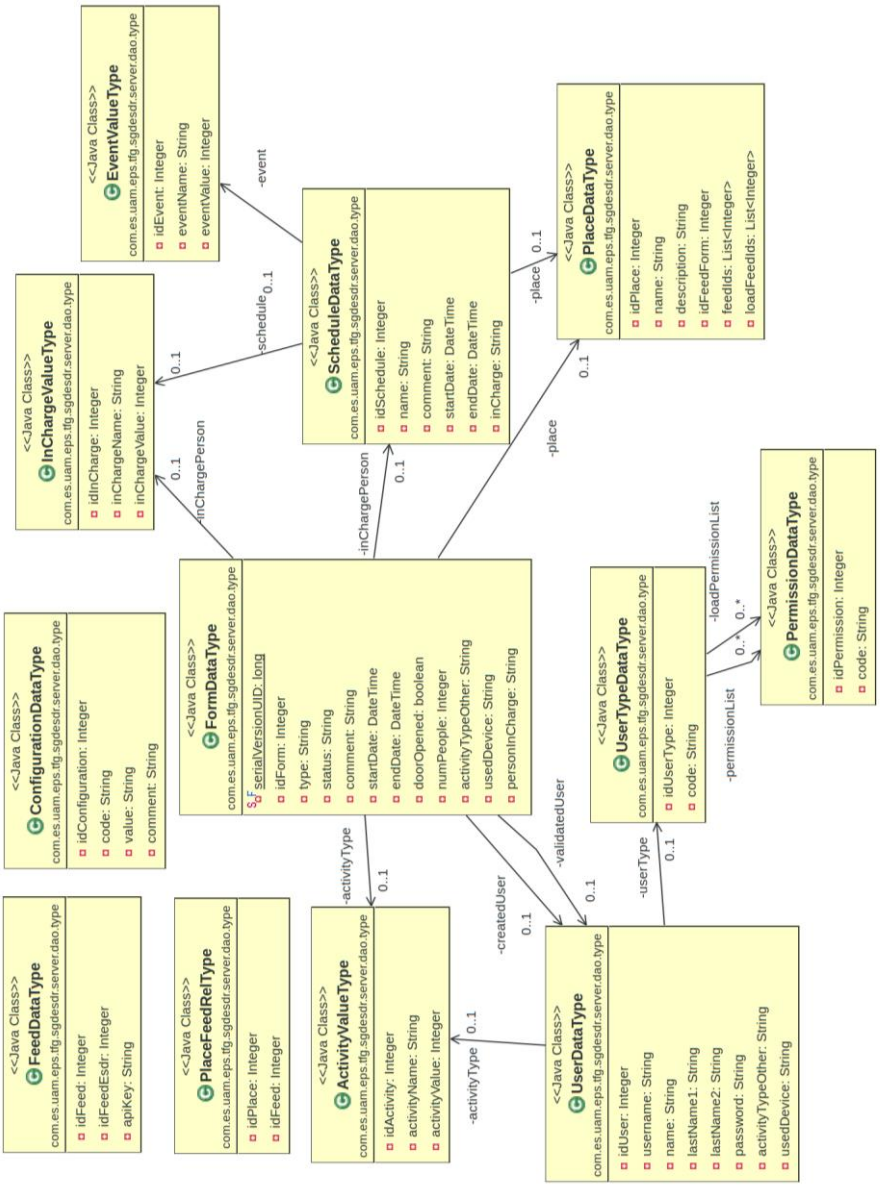


Figura A-1: Diagrama de clases de Type

A continuación se presenta el diagrama del componente ***DAO*** (Figura A-2), que suministra una interfaz común entre la aplicación y la base de datos. Como se puede observar casi todas las clases extienden de *Repository* y *AbstractJpaRepository* y, además, todas las entidades tienen su interfaz *EntityRepository* y su clase *EntityJpaRepository*.

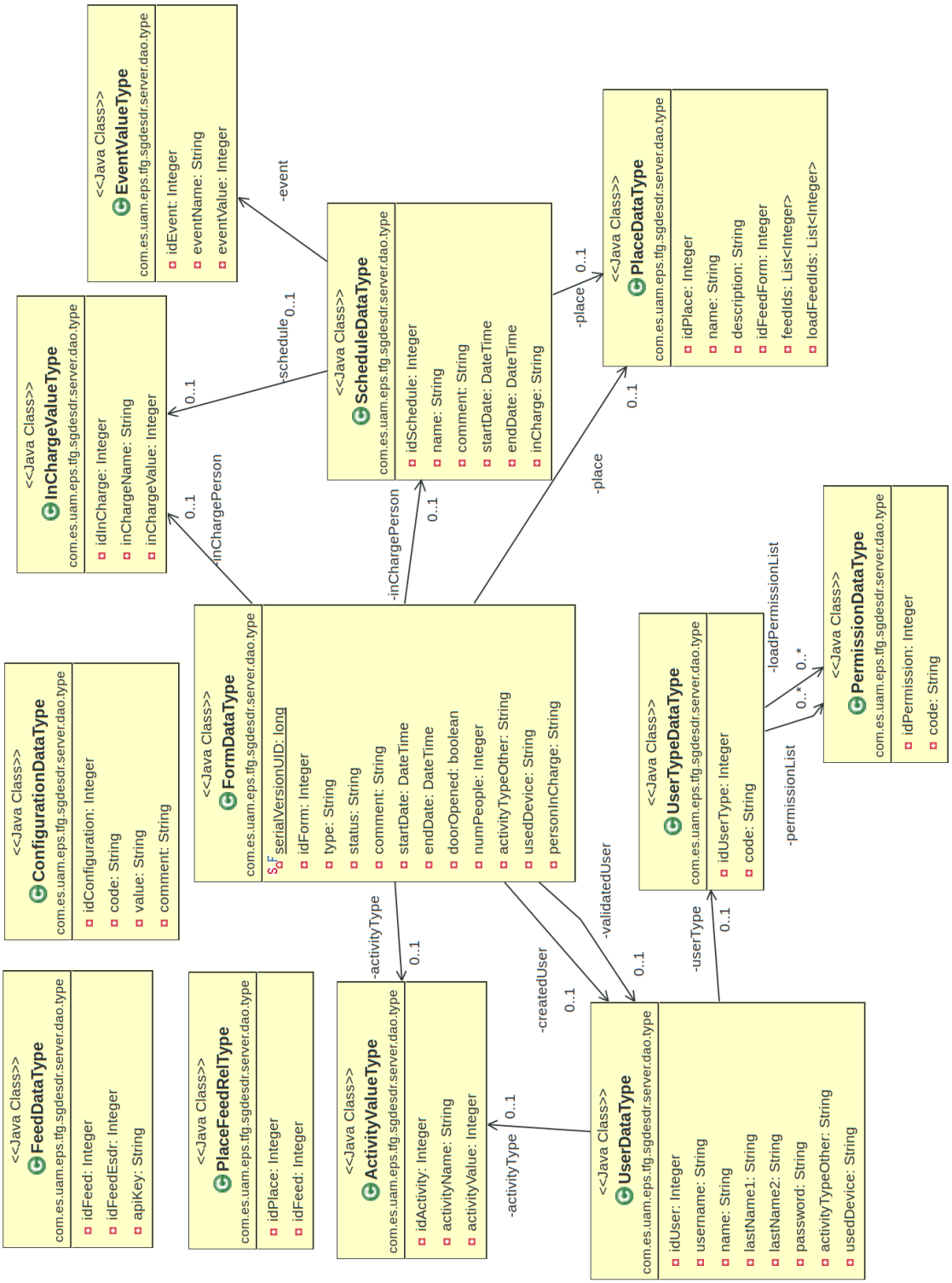


Figura A-2: Diagrama de clases de DAO

Para realizar alguna acción sobre los datos de la base de datos será necesario invocar los métodos de **servicios**. Éstos serán servicios web y se estructurarán tal y como se muestra en las figuras Figura A-3 y Figura A-4. Habrá una clase donde se implementarán los métodos para realizar la conexión con el servicio, una interfaz donde se declararán todos los métodos y una clase donde se implementarán estos métodos.

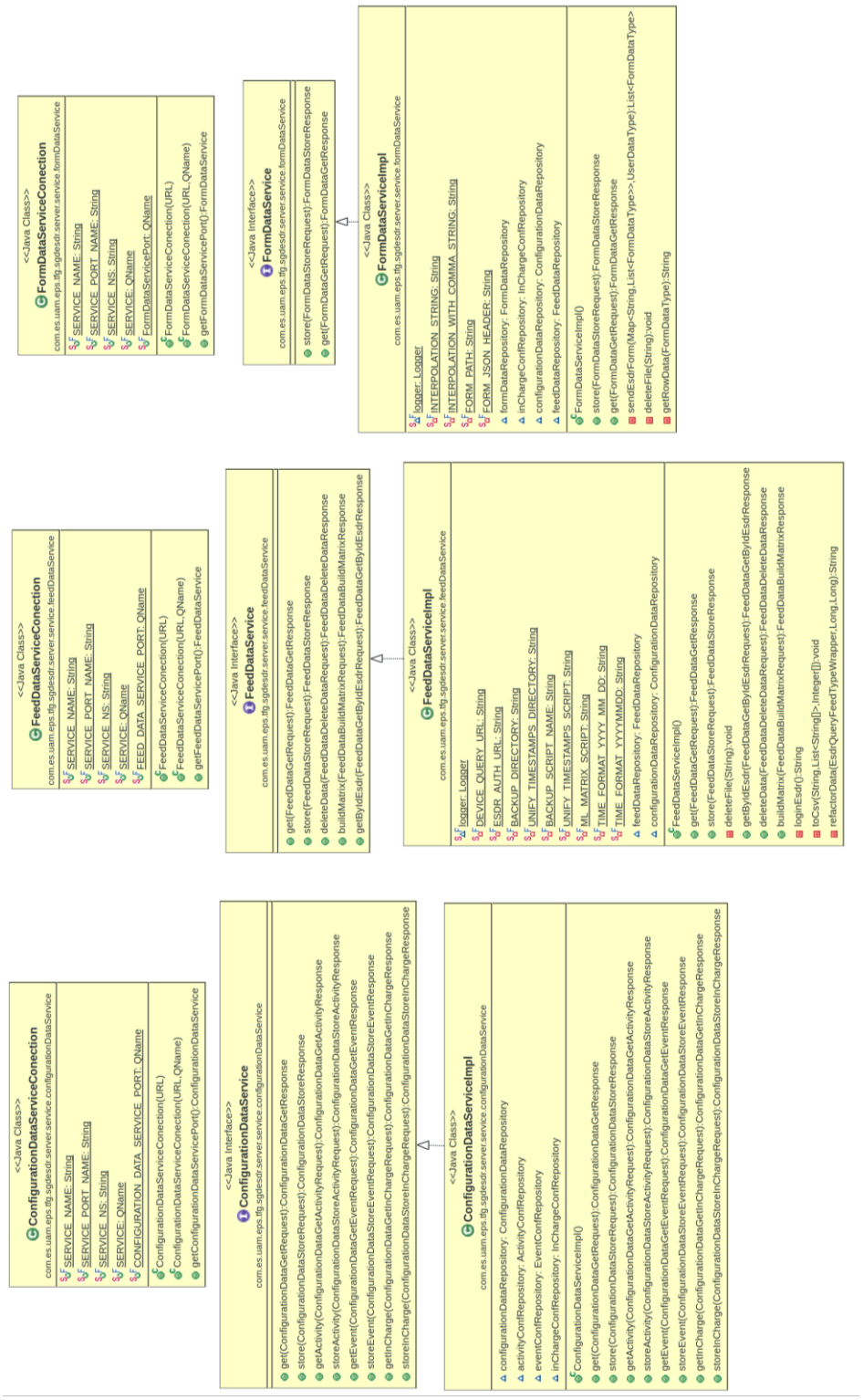


Figura A-3: Diagrama de clases de servicios 1

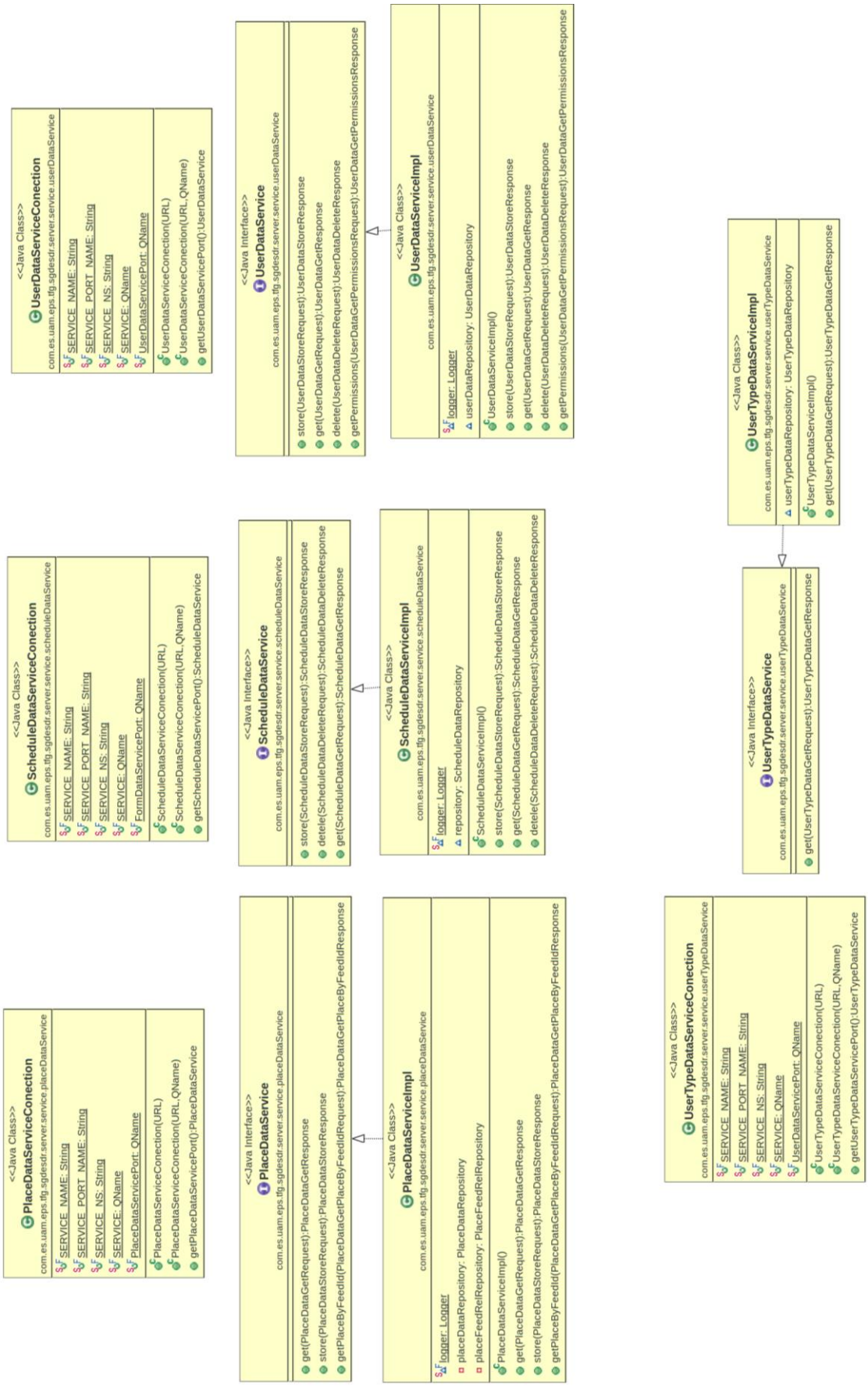


Figura A-4: Diagrama de clases de servicios 2

Los **controladores** se encargarán de recoger las acciones introducidas por los usuarios, y actualizar los modelos y las vistas según dichos eventos. Habrá un controlador por tipo de entidad y un controlador base donde se implementarán los métodos comunes. (Figura A-5, Figura A-6, y Figura A-7).

Figura A-5: Diagrama de clases de controladores 1

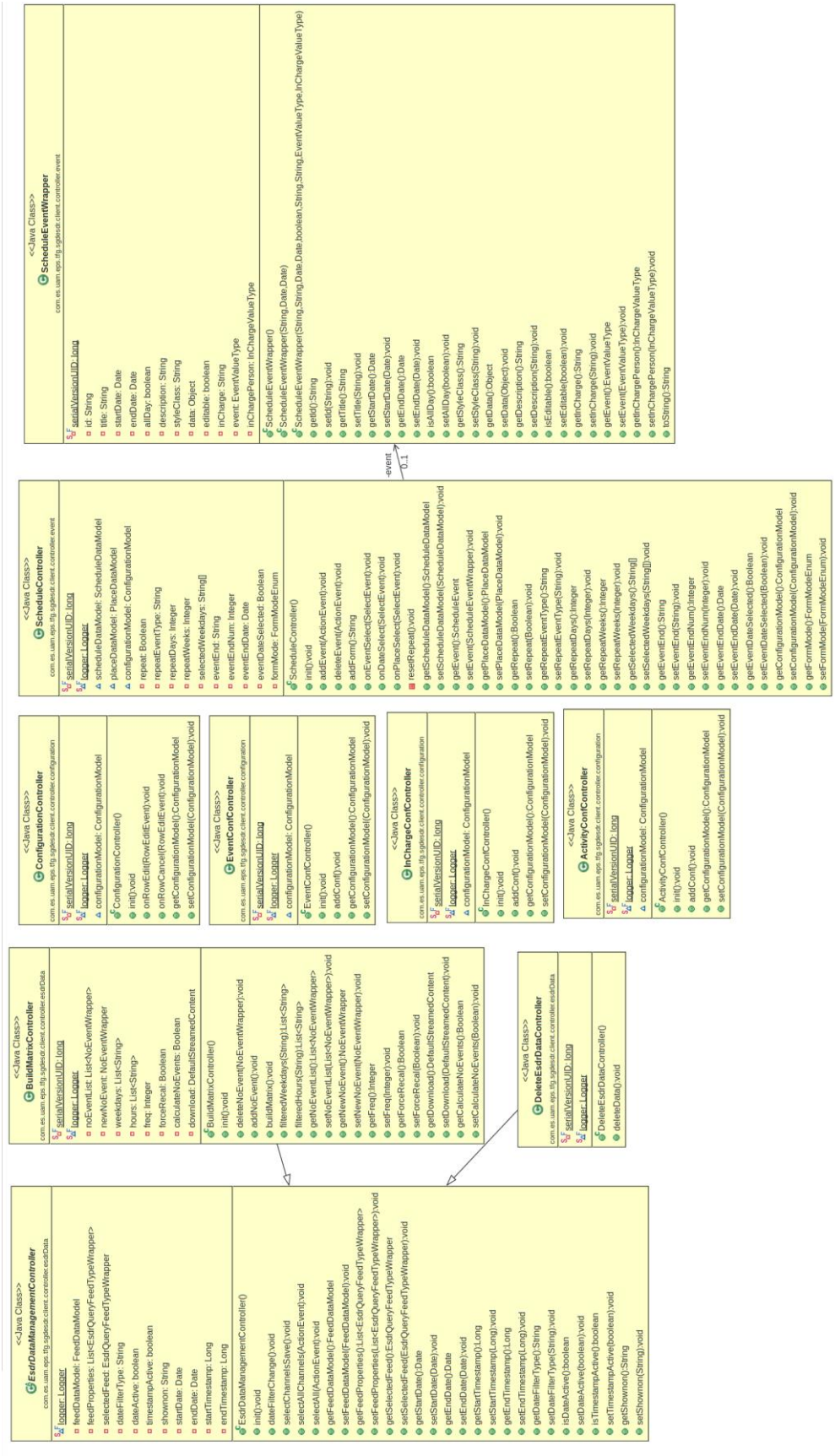


Figura A-6: Diagrama de clases de controladores 2

Figura A-7: Diagrama de clases de controladores 3

Los **modelos** serán las clases que se ocuparán de invocar los métodos para el almacenamiento y recuperación de datos de la base de datos. También habrá un modelo por tipo de entidad y un modelo base donde se implementarán los métodos comunes tal y como se muestran en las figuras Figura A-8 y Figura A-9.

Figura A-8: Diagrama de clases de modelos 1

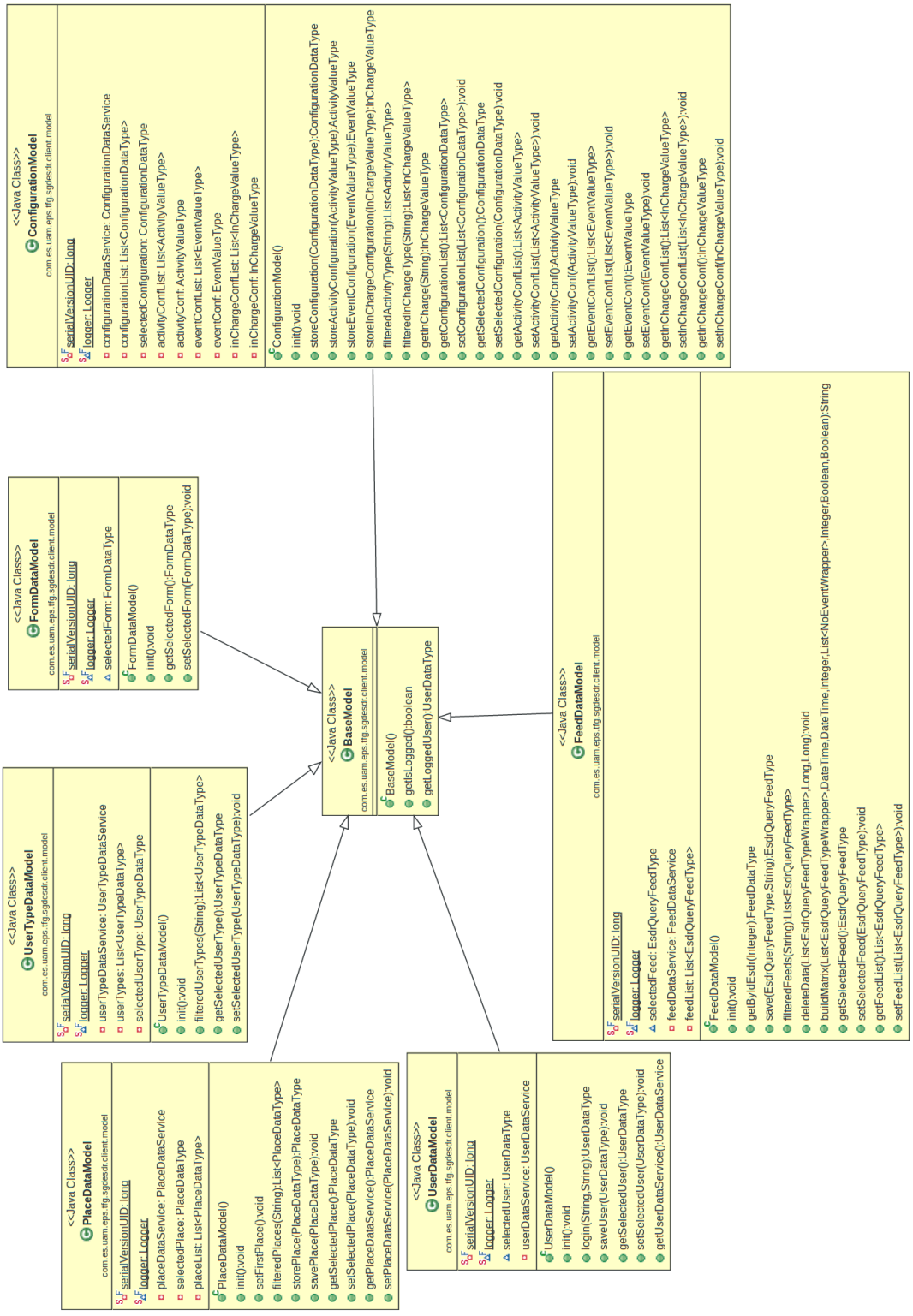


Figura A-9: Diagrama de clases de modelos 2

B. Maquetas

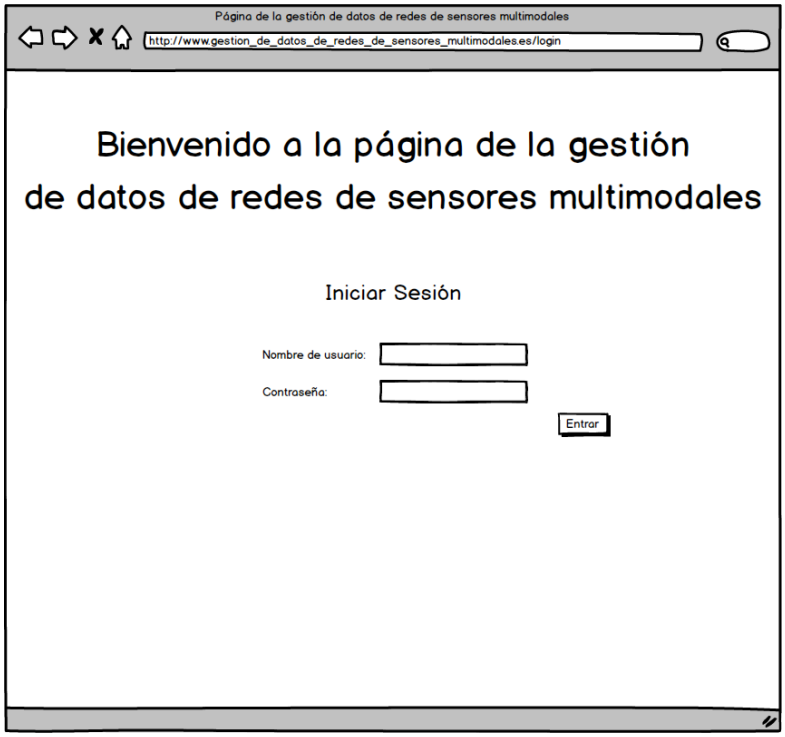


Figura B-10: Maqueta login

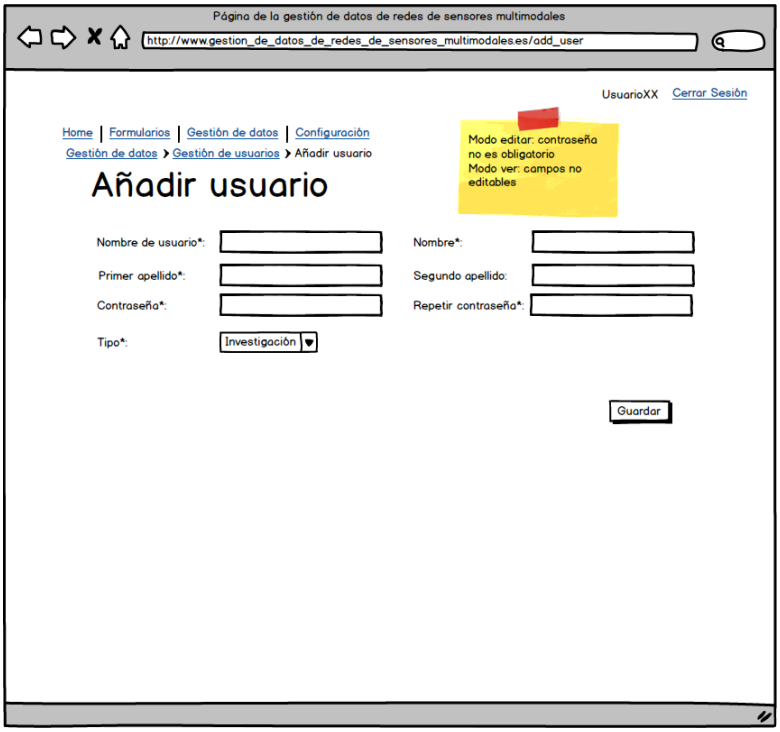


Figura B-11: Maqueta añadir usuario

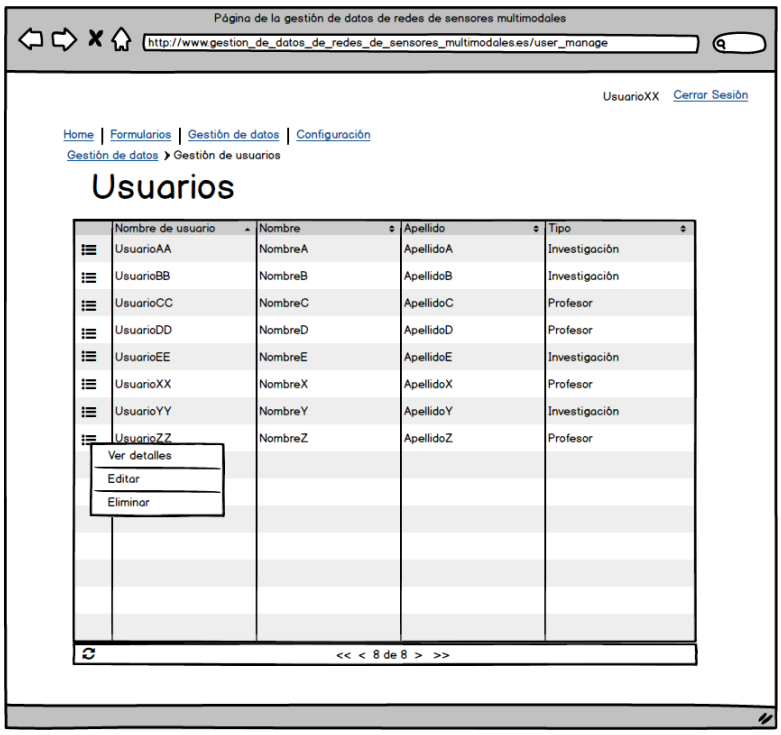


Figura B-12: Maqueta ver usuarios

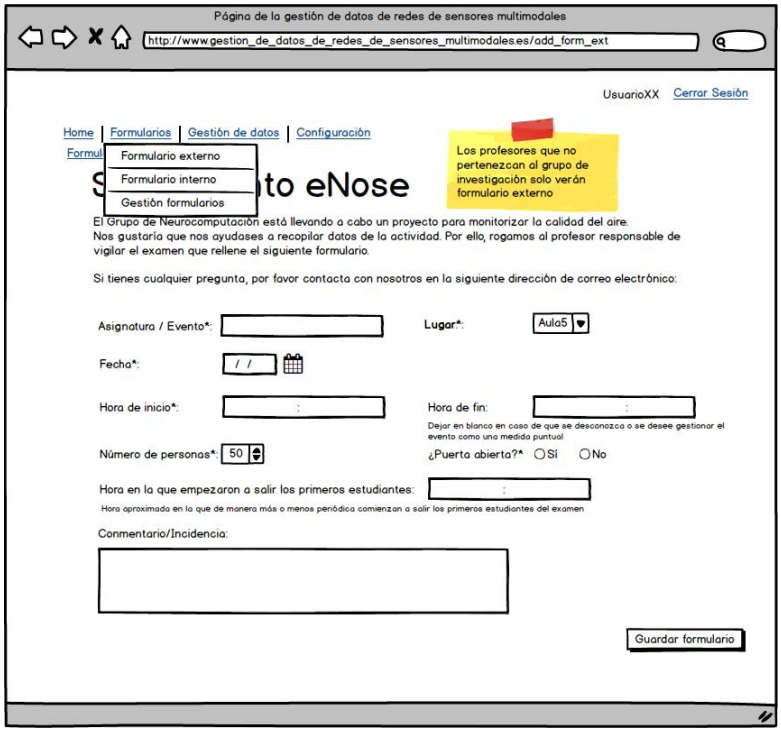


Figura B-13: Maqueta rellenar formulario externo

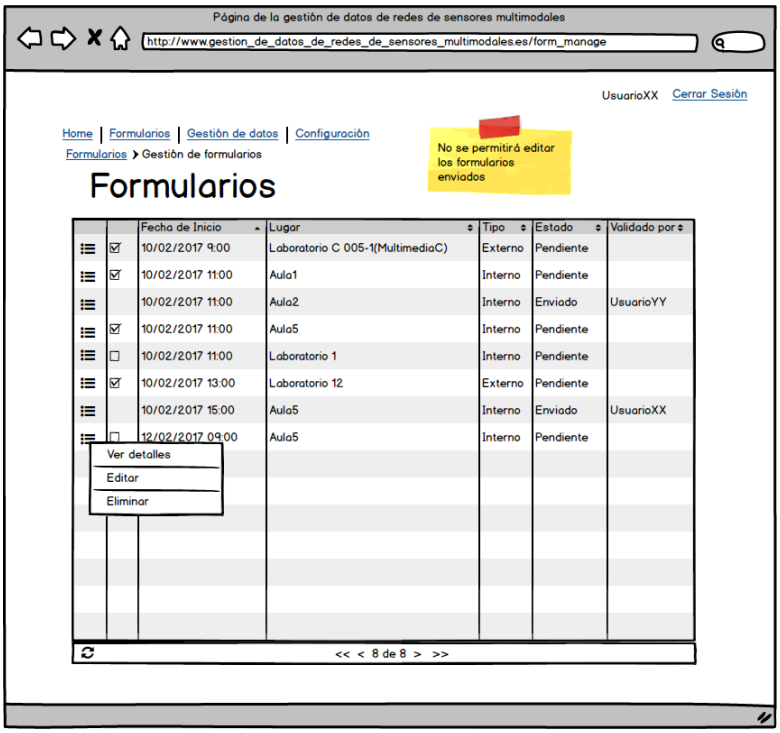


Figura B-14: Maqueta ver formularios

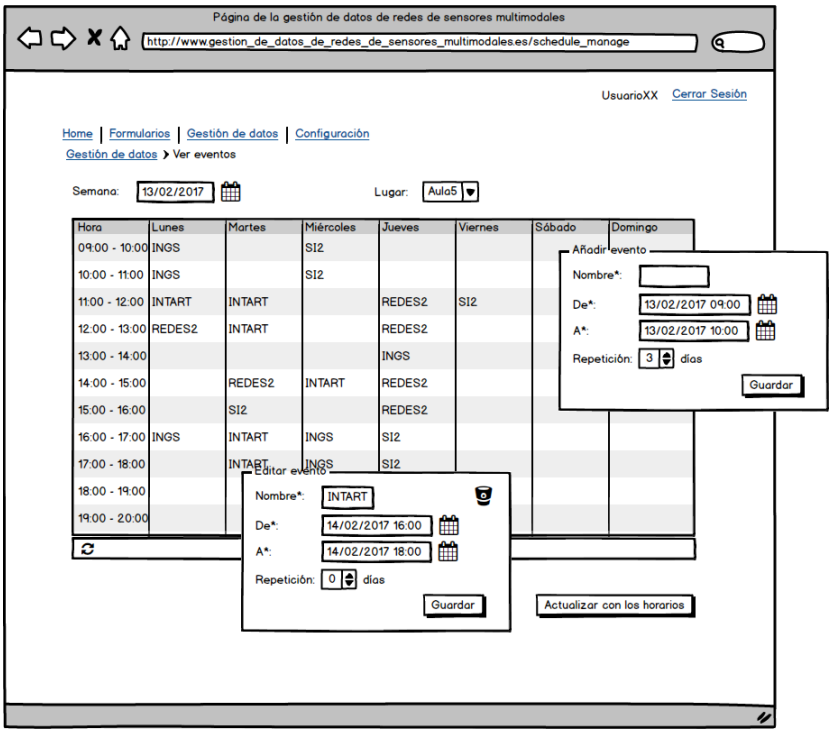


Figura B-15: Maqueta ver eventos

C. Interacción con ESDR con comandos curl

El sistema ESDR tiene un API REST que permite utilizar instrucciones de comandos *curl* u otras herramientas para hacer solicitudes HTTP para obtener datos en ESDR [20].

Para poder utilizar ESDR es necesario crear una cuenta en *esdr.cmucreatelab.org*. Una vez creada y verificada hay que crear un cliente OAuth2 para la cuenta. Es necesario crear un fichero *auth.json* para la autenticación. En este fichero debe tener la siguiente estructura:

```
{
   "grant_type" : "password",
   "client_id" : "my_client",
   "client_secret" : "Secret secret, I've got a secret!",
   "username" : "email@example.com",
   "password" : "bartley"
}
```

Una vez creado el fichero, se debe enviarlo al ESDR instalado con el comando:

```
curl -X POST -H "Content-Type:application/json" https://esdr.cmucreatelab.org/oauth/token -d @auth.json
```

Y se obtendrá una respuesta del tipo:

```
{

"access_token":"5ea621c52a9eff664d6dec7ce4035b33d4712ed69a945521e73c8f40a305fe18",

"refresh_token":"64cbf6915fbd25bf6954f807332810a316ad3932526f7e8acdef742ce8ef11c3",
   "userId":2,
   "expires_in":604800,
   "token_type":"Bearer"
}
```

Se debe guardar el valor de *access_token* al ser necesario para realizar operaciones de escritura en ESDR. Este valor se expira cada semana, es decir, si se quiere realizar operaciones de escritura, será necesario hacer esta consulta cada semana.

A continuación se da un ejemplo de un fichero JSON para crear un producto. Se puede dejar *defaultChannelSpecs* vacío.

```
{
   "name":"my_test_product",
   "prettyName":"My Test Product",
   "vendor":"Acme, Inc.",
   "description":"A sensor that senses stuff.",
   "defaultChannelSpecs":{
      "version":1,
      "channels":{
         "temperature":{
            "prettyName":"Temperature",
            "units":"C",
            "range":{
               "min":-273.15,
               "max":null
            }
         },
         "battery_voltage":{
            "prettyName":"Battery Voltage",
            "units":"V",
            "range":{
               "min":0,
               "max":5
            }
         }
      }
 }
}
```

Y para registrar el producto se usa el comando:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
Bearer                                          ACCESS_TOKEN_HERE"
https://esdr.cmucreatelab.org/api/v1/products -d @product.json
```

Para crear un dispositivo usar el comando:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
Bearer                                          ACCESS_TOKEN_HERE"
```

```
https://esdr.cmucreatelab.org/api/v1/products/PRODUCT_ID/devices -d @device.json
```

Y el fichero JSON debe contener:

```
{
   "name" : "Widget 2000",
   "serialNumber" : "abcdefghij0123456789"
}
```

Por último, para crear un *feed* usar:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: Bearer ACCESS_TOKEN_HERE" https://esdr.cmucreatelab.org/api/v1/devices/DEVICE_ID/feeds -d @feed.json
```

El fichero JSON debe contener:

```
{
   "name" : "Back porch",
   "exposure" : "outdoor",
   "isPublic" : 0,
   "isMobile" : 0,
   "latitude" : 40.443403,
   "longitude" : -79.94564
}
```

Otros comandos útiles como cargar los datos a ESDR

```
curl -X PUT -H "Content-Type:application/json" https://esdr.cmucreatelab.org/api/v1/feeds/FEED_API_KEY_HERE -d @data.json
```

El fichero de datos debe tener esta estructura:

```
{
   "channel_names" : ["temperature", "conductivity", "battery_voltage"],
   "data" : [
      [1380276279.1, 19.0, 516, 3.85],
      [1380449602, 19.2, 485, 3.84],
      [1380472357, 18.6, 485, 3.84],
```

```
            [1380556690, 18.3, 501, 3.84],
            [1380643808, 19.5, 583, 3.84],
            [1381154132.009, 18.5, 565, 3.84]
        ]
    }
```

Exportar datos:

https://esdr.cmucreatelab.org/api/v1/feeds/FEED_ID_OR_API_KEY/channels/ONE_OR_MORE_CHANNELS_COMMA_DELIMITED/export?from=UNIX_TIME_SECS&to=UNIX_TIME_SECS&format=[csv|json]

Todos los *feeds* del sistema están disponibles en la siguiente URL.

https://esdr.cmucreatelab.org/api/v1/feeds

Para más detalle, consúltese https://github.com/CMU-CREATE-Lab/esdr/blob/master/HOW_TO.md

D. Manual de instalación

Para poder instalar esta aplicación es necesario disponer de tres herramientas:

- Java 8, que se podrá descarga de la página de Oracle o instalar desde la línea de comandos.

- Wildfly 10: los pasos a seguir son:
  1. descargar wildfly 10
     $ wget http://download.jboss.org/wildfly/10.0.0.Final/wildfly-10.0.0.Final.tar.gz
  2. instalar en /opt
     $ tar -xvzf /xx/wildfly-10.0.0.Final.tar.gz
  3. cambiar los permisos
     $chmod -R 755 wildfly-10.0.0.Final/
  4. ejecutar
     $ ./wildfly-10.0.0.Final/bin/standalone.sh
  5. entrar http://127.0.0.1:9990/error/index.html para ver si se encuentra ejecutando
  6. crear usuario en wildfly
     $./wildfly-10.0.0.Final/bin/add-user.sh
     admin / admin, todo yes
  7. entrar en consola de wildfly http://127.0.0.1:9990/console con admin / admin

  Con estos pasos será suficiente instalar y ejecutar wildfly. Y seguro que en Google también se podrá encontrar muchos tutoriales para instalarlo.

- Un base de datos de MySql, se podrá instalar desde la línea de comandos.

También se podrá instalar Maven si se quiere compilar el código.

Tanto el código fuentes como los artefactos del proyecto se encuentran todo en el repositorio https://bitbucket.org/qwertyue/tfg-sistemagestiondatosesdr .

Para instalar la aplicación es necesario crear una base de datos en MySql, bastaría con ejecutar los scripts *~/código/data/scripts/ddl.sql* y *dml.sql* del mismo directorio con el objetivo de precargar los datos necesarios para el correcto funcionamiento de la aplicación. En el fichero *dml,sql* se encuentra un usuario Admin de tipo INT, que permite acceder a la aplicación, si no se tiene ningún otro usuario.

Una vez preparado la base de datos, es necesario desplegar en el directorio *deployments* de Wildfly (normalmente es *~/wildfly-10.0.0.Final/standalone/deployments/*) el fichero .ear que se encuentra compilado en el directorio *~/codigo/SGDEsdr/ear/target/ear-1.0.ear* del repositorio. Desde la consola de Wildfly también se puede subir desplegables.

Wildfly proporciona un fichero log *~/wildfly-10.0.0.Final/standalone/log/server.log* donde suele haber mensajes informativos sobre el estado actual del servidor y de la aplicación. Estas informaciones suelen ser muy útiles.

Si se quiere compilar el código, es necesario disponer de Maven. Una vez instalado Maven, ejecutar *mvn clean install* en el directorio *~/codigo/SGDEsdr/* para compilar todo y generar el fichero .ear (se genera en *~/codigo/SGDEsdr/ear/target/*).